ICA Workshop on Generalisation and Multiple representation- 20-21 August 2004 - Leicester

Integration of Agent-based Generalisation with Mainstream Technologies and other System Components

by Dieter Neuffer, Tony Hopewell and Peter Woodsford

Laser-Scan, Cambridge, UK

dieter.neuffer@laser-scan.com
tony.hopewell@laser-scan.com
peter.woodsford@laser-scan.com

Abstract

The paper starts by presenting the case for a two stage approach of model and cartographic generalisation. It outlines the advantages of the use of active agent technology and describes a mature implementation (the Clarity product) which uses mainstream technologies to provide an extensible and user-friendly system. The task does not end with the initial generalisation process - maintenance is also a key issue and the facilities for incremental generalisation are described. The system is not a closed environment, and the paper concludes with how it can be integrated with other system components.

1 Introduction

Generalisation is an extremely complex process. It has been analysed extensively in literature, see e.g. [Kilpeläinen 1999], and lot of research has been done, both theoretically and from a practical point of view. The overall process can still not be automated fully but great progress has been made in recent years. Production systems that integrate automatic and manual generalisation are now becoming available and can achieve a large reduction in production costs, as described in [Jahard 2003] and elsewhere in this workshop.

The overall generalisation process can be split up and its constituent parts can be classified in many different ways. We use the concepts of model generalisation and cartographic generalisation. In this paper, the term model generalisation is used in the sense of generalisation from a digital data model of high resolution to a digital data model of a lower resolution. See [Schürer 2002] for an in-depth discussion of model generalisation in this sense.

Using a two stage approach, the scale change is performed by model generalisation and is done completely at the data model level without taking into account any representation. This leaves the cartographic generalisation to deal with the correct symbolisation and the placement of the cartographic objects and the labels on the map. This task has to balance many conflicting objectives. Introducing model generalisation as a 'pre-process' results in a welcome simplification as well as generating useful data products.

1.1 Model Generalisation

The role of model generalisation is to reduce to amount of data to the level suitable for the target scale. The main objectives of are:

- Data volume reduction
- Removal of detail that is not required for the target scale
- Data harmonisation

Often the data forming the starting point of model generalisation was recorded over a considerable period of time, and maybe by different organisations. In such a situation the uniformity of the data cannot always

be guaranteed. As a result, data harmonisation is required to ensure that the variation in the source data does not have any negative effects on the quality and uniformity of the target data.

In general, model generalisation consists of the following processes:

- Selection of objects by feature class
- Individual selection of objects by attribute value and/or context
- Geometry-type changes (area to point, area to line, line to point)
- Typification (optional)
- Geometric simplification (e.g. using the Douglas-Peucker algorithm)

Note that none of the above operations displace objects - hence the data remains in its correct location as is generally required for data products. These processes are not independent of each other and need to be carried out in a carefully designed sequence. In an implementation using a Laser-Scan system they are translated into a series of processes sequences. Each process sequence consists a number of process methods. This approach can process large amounts of data. Typically a whole state will be held in one seamless database. The processing will then be done object by object across the whole dataset. Intermediate checkpoints ensure that backtracking is always possible.

The on-line presentation [AdV 2003] shows some of the results of applying such a sequence of processes. It also gives a good overview of the ATKIS DLM50 model generalisation project.

1.2 Cartographic Generalisation

Cartographic generalisation, when applied as a second process after model generalisation, deals with the best use of map space for a specific map representation. The objective is to optimise legibility at a given scale and for the purposes of the map. This includes the following processes:

- Applying map-specific representations
- Displacement to deal with legibility issues
- Label placement

Laser-Scan uses active agent technology for cartographic generalisation, as discussed in detail below. Unlike the conventional approach, which is based on the design of a suitable sequence of processes, this approach starts with the definition of the desired outcome. The ability to define the appearance of the finished map is a key advantage of the agent approach. This is done by creating a 'map specification'. Processes still need to defined - but it is the agent system that selects the best processes for a given object, taking into account the context. This technique also works across feature classes - for example, the generalisation of a road takes into account the position of a building nearby - and vice-versa.

Two definition stages are required:

- Define the map specification - this is where the requirements for the finished map are specified.
- Define the processes available for generalisation - and the order in which they will by tried by the agent system. The system decides which processes are best suited for any given situation.

Laser-Scan's Clarity provides a modern Java interface for the specification of processes and uses the XML format for map specifications. XML is also used for transfer parameter encoding in the context of model generalisation.

1.3 Text Placement

For text placement the same agent technology can be used as for cartographic generalisation. This approach has been realised in Laser-Scan's ClearText product. The agent technology is ideally suited for label placement - all the concepts used in cartographic generalisation can be re-applied. Additional functionality is provided, for example the use of an abbreviation table for labels, which is used if the space restrictions cannot be resolved in any other way. Using the same agent framework for both cartographic generalisation and label placement allows close integration.

2 Active Agent Technology

Good generalisation requires:

- Contextual analysis - you can't generalise one map feature at a time in isolation. You have to consider groups of objects as a whole.
- Adaptive processing - you can't apply a single algorithm to all features (even of a single class). You have to choose appropriate algorithms according to the circumstances of that feature.
- Backtracking - you can't get it right first time every time. You have to be prepared to assess whether an operation has made things better or not, and be prepared to undo it and try something else.

2.1 AGENT Project

The AGENT (Automated GEneralisation New Technology) research project [Ruas 2000] was initiated as an EC funded project (Esprit 24939) in the late 1990s. Laser-Scan joined the multi-national consortium as the software supplier. Other members included IGN (the French national mapping agency in Paris) in the lead role, and three universities: Zurich and Edinburgh (for their expertise in geography and cartography) and Grenoble (for its work in artificial intelligence). Three years of intensive research and development resulted in a prototype agent-based generalisation system, based on Laser-Scan's Gothic LAMPS2 object-oriented software.

2.2 Concepts

Agents act as self-aware active software objects that co-operate, subject to a set of constraints, to achieve a goal. Two levels of generalisation agents are employed:

- **Micro-agents**: agents that monitor, evaluate and propose planned courses of action on specific map objects such as individual buildings.
- **Meso-agents**: dynamically delimitated areas that control the relationships between groups of map objects with themselves and other map objects. A meso-agent for example could control all buildings found in an area encircled by roads.

Other key concepts are:

- **Constraints**: each agent has a set of constraints to guide its generalisation behaviour. Each class of constraint normally controls one characteristic of a particular class of agent, e.g. the squareness of a building.
- **Measures:** to detect generalisation conflicts: e.g. bend coalescence, overlapping symbols, oblique junctions.
- **Algorithms:** for simplifying forms, caricaturing bends (exaggerating or deleting), moving elements (roads or churches), maintaining coherence between themes.
- **Plans** are proposed by a constraint in order to improve the particular characteristic of the agent monitored by that constraint. Plans consist of one or more algorithms with specific parameters.

Both micro and meso-agents strive to improve their relative 'happiness' according to a set of constraints. The interface for defining these multi-level constraints is discussed below in Section 3.

2.3 Agent Lifecycle

Based on the experience with the AGENT project, the agent lifecycle has been completely redesigned. An overview of the new agent lifecycle is shown in Figure 1. This lifecycle has the following properties:

1) it handles the life of an agent, from activation to passivity;
2) it invokes constraints (possibly with associated measures), plans and algorithms as needed;
3) it is implemented using object methods (behaviours);
4) it is highly customisable by the overriding of methods;
5) it implements 'hill-climbing' algorithms to explore possible states from execution of plans, and choose the best;
6) it can preserve history of state for subsequent analysis of how and why;
7) it has breakpoints for debugging and investigation;

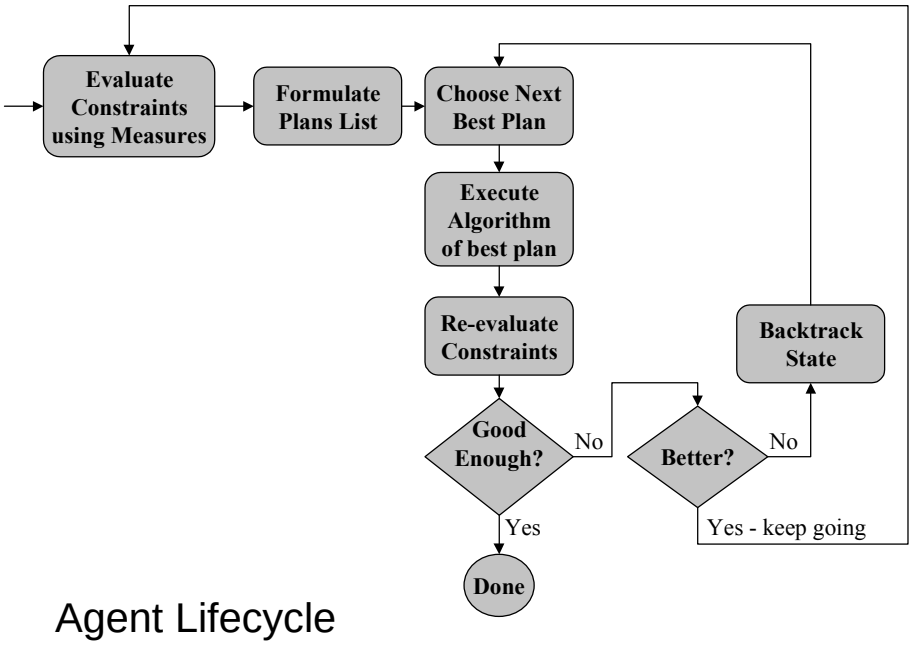

Figure 1: Agent Lifecycle - how to find a good generalisation solution automatically

3 Clarity

Laser-Scan's Clarity generalisation product uses the new agent lifecycle. It provides a modern Java interface for the specification of processes and uses the XML format for map specifications, and in the context of model generalisation XML is used for transfer parameter encoding. The application consists of a number of Java clients:

3.1 Java client for generalisation definition

Java interfaces are provided to setup the parameters for generalisation processes. This includes the setting up the constraints, as shown in Figure 2. The Java-based interface also includes the setting of global and class specific parameters for the agent methods, algorithms and constraints. The interfaces are able to save and load the parameters to and from XML files.

Generalisation historically required a large number of user-definable parameters. In the agent-based scenario, many of these parameters are superseded by a 'map specification', defining what is to be achieved, coupled with a set of 'constraints' which put limits on allowable change. In either case, there is a need to define and store a complex set of definitions.

XML is the format of choice because it allows these definitions to be structured in a logical way. It can be read by humans, and displayed and edited in a large number of applications, which makes it easy to check that no accidental errors are introduced. This makes XML an ideal depository for parameters that are required by the agent system.

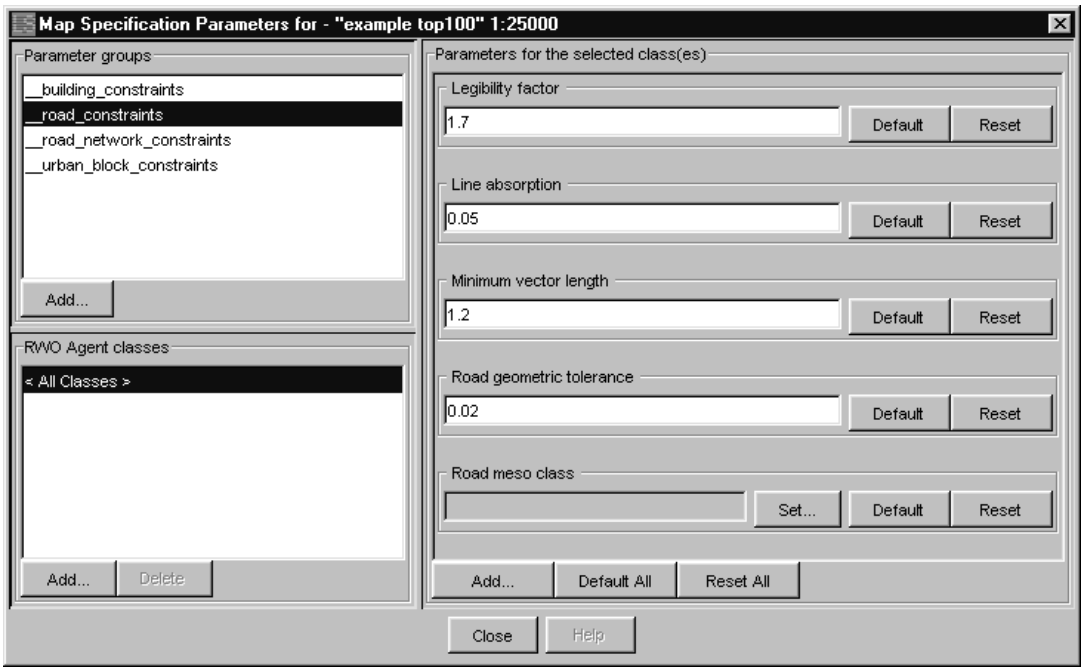

Figure 2 - Java interface for generalisation definition - algorithms and classes

3.2 Java client for generalisation execution

Two interfaces are provided regarding execution of generalisation processes:

- The Agent Process Diagnostic Interface is an interactive tool for setting up, manipulating and inspecting the agent stack. It is of use primarily for flowline developers or researchers.
- The Process Job Interface is for running sequences of process methods with parameters, including object selection criteria. Invocation of agents is done by a particular process method, taking the map specification as a parameter.

3.3 Java client for interactive completion

The JADE environment can also be used for visualisation of generalisation results and for interactive completion of generalisation for the cases where human intervention is needed. The following screenshot (Figure 3) shows the JADE user interface, displaying the results of some experiments in agent-based generalisation of buildings.

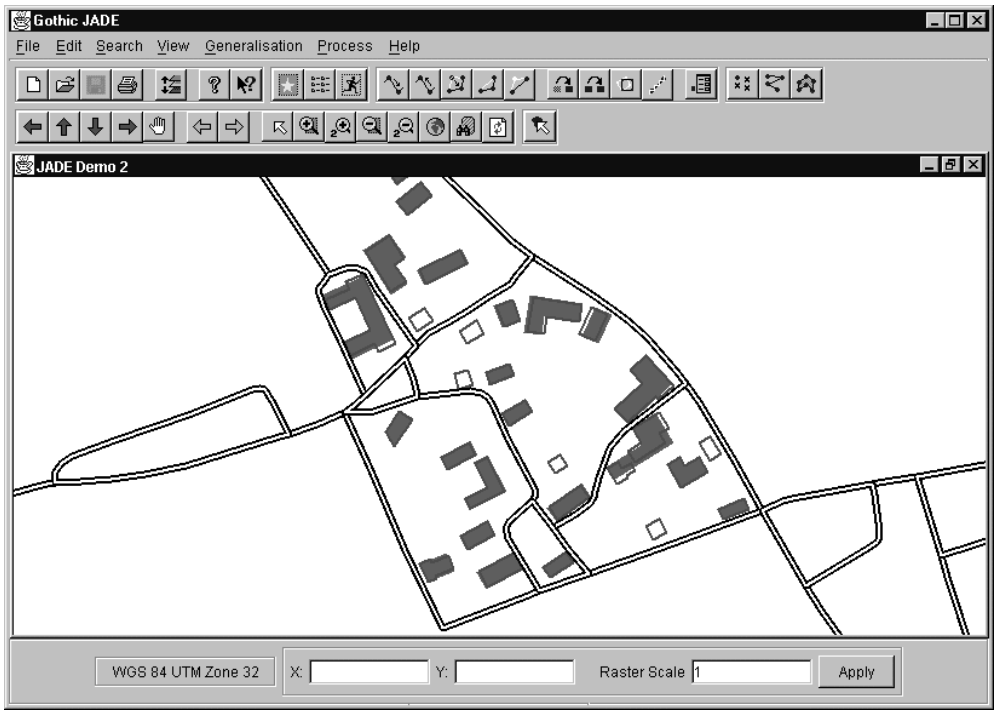

Figure 3: Java interface for interactive completion

3.4 Extensibility

Clarity comes with set of pre-defined constraints, measures and algorithms. New measures, constraints and algorithms for Clarity can be implemented in Java. As the development continues, new algorithms can be added. Significantly, they can also be added by users. All that is required apart from the knowledge of generalisation and the algorithm is some basic knowledge of Java. As and when as algorithms are improved or new ones become available, they can be added to the framework. This evolutionary approach allows new research to be conducted and results incorporated as appropriate.

4 Incremental Generalisation

The derivation of generalised data is only one aspect of the overall system. Maintenance is another aspect, of rapidly escalating importance as demand for consistent and up-to-date data grows. Arguably it, the most important aspect of a production system. It is also the most complex process. A lot of research has been done on this aspect of generalisation; more information on this subject can be found in [Badard 1999], [Harrie 1999] and [Kilpeläinen 1995].

Particularly in the case of cartographic generalisation, the situation gets more complex because of the need to deal with manual edits. These need to be preserved wherever possible so they can be re-applied in order to avoid the cost of repetition and to retain quality and consistency. Without this degree of automation there is no guarantee that manual edits will be done in exactly the same way if they have to be repeated. A cost-efficient solution needs to only re-generalise what is required as a result of a change in the source data. It also needs to assist with the re-application of the manual edits.

Why is incremental generalisation so complex? Fundamentally, because generalisation is a context dependent task. So given a set of updated objects, it is clearly not sufficient to just re-generalise those. The

generalisation of other objects may either be affected by the changed objects, or, vice-versa affect these. Hence these other objects also need to be re-generalised.

There are two typical update scenarios. In the first scenario, there are few isolated clusters of changes, e.g. a new road has been built or a residential area has grown. In the second scenario the changes are more evenly spread over the whole dataset. Such a situation occurs if new data becomes available (attribute values or geometry updates) for certain feature classes.

For best performance, these scenarios require tailor-made approaches.

The strategy for the first scenario is:

- The most important requirement is to determine which set of objects ('Area of Influence', AoI) needs to be re-generalised for any given update. This is done using parentage information in the form of bi-directional references and also relationships with other objects
- An iterative approach can be used to establish the optimum AoI.
- The re-generalised AoI is patched into the main dataset.
- A system to manage manual edits and assist with re-application of these edits.

For further detail on this strategy and diagrams see [Neuffer, 2004].

The approach adopted depends on the fact that, while one feature in the dataset may interact with many surrounding features in a complex way, the degree of interaction declines with distance. If a feature is generalised in the context of a large enough buffer of surrounding features, it will be generalised in the same way as it would be generalised in the context of the entire dataset.

Figure 4 shows how a 'Buffer of unchanged objects' is used to establish the optimum AoI. After the generalisation of an update, an area some distance from the changed objects is expected to be unaffected. This is called the 'Buffer of unchanged objects'. If this buffer turns out to be unchanged then the AoI was chosen large enough. If not, then the AoI needs to be enlarged and the incremental generalisation needs to be rerun. The outermost area contains edge effects. This area is expected to generalise differently not because of the changed objects in the AoI, but because only a part of main dataset is generalised and so the objects at the edge of this area are not within their correct context. Hence the objects in this area are not patched into the main dataset.

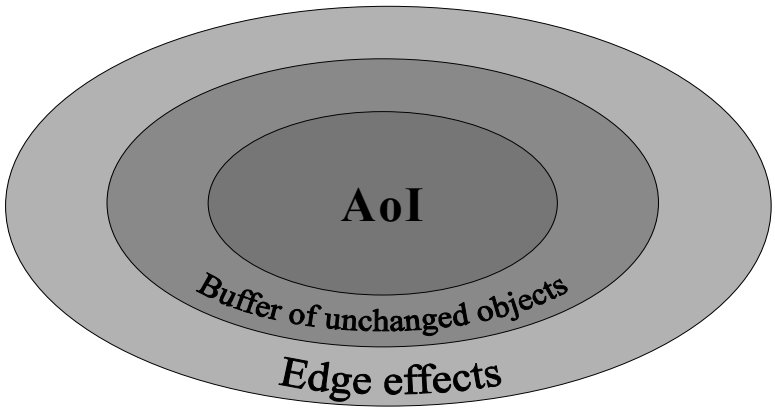

Figure 4: Area of Influence - changes in the source data can have effects some distance away

A powerful system is required for the implementation of automatic incremental updates. The following functionality must be available to meet the requirements discussed above:

- The ability to hold both source and target data models in one dataset. This is needed to build and store parentage information in the form of references. Retrieval of parentage information is a key requirement to determine the AoI. It is important that this information can be retrieved quickly and efficiently - hence bi-directional references are the best solution.
- The use of persistently stored topology. This allows the fast and efficient retrieval of neighbourhood information - another requirement for efficient determination of the AoI. Without topology, spatial searches are required and these are significantly slower.

For the second scenario, i.e. more evenly distributed changes, it is quite possible that the above strategy will produce just one large cluster of changes containing most of the dataset. If this is likely, then performance benefits can be obtain by omitting the process to determine AoI, and instead re-generalise the whole dataset.

Two points are worth noting when using this strategy. Firstly, the re-generalised data still needs to be patched into the main dataset to ensure that only changed objects are updated. And secondly, a system to assist with the application of previous manual edits is required in the same way as in the first scenario.

5 Open Environment

The previous sections have highlighted that Laser-Scan's generalisation solution offers the full integration of all the stages required for a complete generalisation solution:

- Model generalisation - using Gothic process methods.
- Cartographic generalisation - using Laser-Scan's Clarity product.
- Label placement - using either Laser-Scan's ClearText product, a third party product, or a customer implementation.
- Interactive editing and completion.

At the same time, the system is not only open for modifications by the user, but it also makes the required setting up and tuning as easy as possible. This is done using industry standard tools and formats:

- XML for the definition of parameter files and map specifications.
- Java for writing user-specified process.
- Modern Java user-interface for setting up the generalisation processes.

Laser-Scan's open approach goes further. It can be integrated easily with third party software. Three significant examples can be cited of this approach:

5.1 Use of Oracle Spatial as Data Repository

KMS Denmark have developed a workflow in which Oracle Spatial is used as the data repository for both topographic framework data (Top10DK) and product data derived by generalisation using Clarity [West-Nielsen, 2001]. The data transfers required are accomplished using Safe Software FME. This database-centric workflow also allows flexibility of choice of editing software for the framework data maintenance and of cartographic finishing software (Intergraph Geomedia).

5.2 Use with WinPat Text Placement Software

WinPat is a label placement program developed by the IGN France Carto2001 project based on the research carried out at the COGIT Laboratory [Lecordix 2004]. WinPat takes the data from a database in the form of text files and places it as: horizontal text, straight oriented text or curved text. WinPat can be used for data at any scale; it has already been used for maps at 1:25,000, 1:50,000, 1:100,000, 1:250,000 and 1:1,000,000. WinPat can take account of a wide range of cartographic criteria. It can be configured by the user to suit the user's cartographic preferences. WinPat output is in the form of text files that can be imported into various geographic information systems. It has already in use with Laser-Scan's LAMPS2/Clarity.

6 Conclusions

The distinction between model and cartographic generalisation is a useful one for both operational reasons and to reduce complexity to manageable proportions. Active agent technology is now a valuable and mature approach to generalisation, particularly for cartographic generalisation. The new implementation of the agent lifecycle based on XML and Java makes the Clarity product a powerful and extensible environment for both research and use in production. In the later context, viable procedures for managing update by incremental generalisation are as important if not more so than the initial generalisation procedures. An open approach to integration with other system components is bringing the benefits of the active agent technology for generalisation to a wide range of users in a variety of flowlines.

7 References

AdV InterGeo Presentation, 2003:
http://www.adv-online.de/neues/intergeo_vortraege/pdf/19_freitag/generalisierung_fr_19.pdf

Badard, T. and C. Lemarié, C. ,1999. Propagating updates between geographic databases with different scales. Atkinson, P. and D. Martin, (Eds.), Innovations in GIS VII: *GeoComputation*, Chapter 10, Taylor and Francis, London.

Harrie, L., and A.-K. Hellström, 1999. A Prototype System for Propagating Updates between Cartographic Data Sets. *The Cartographic Journal,* Vol. 36, No. 2, pp. 133-140.

Jahard, Y., Lemarié, C. and Lecordix F., 2003, The Implementation of New Technology to Automate Map Generalisation and Incremental Updating Processes, *Proceedings of the 21st International Cartographic Conference (ICC)*, Durban, South Africa, ISBN: 0-958-46093-0.

Kilpeläinen, T. and T. Sarjakoski, 1995. Incremental Generalization for Multiple Representations of Geographical Objects. Müller, J-C, J-P Lagrange and R. Weibel (eds), *GIS and generalization,* Gisdata 1, ESF, Masser, I. and F. Salgé (series eds.), Taylor & Francis, pp. 209–218.

Kilpeläinen, T., 1999. Research on Generalisation. *Report of the Finnish Geodetic Institute*, Vol 6: Map Generalisation in the Nordic Countries, pp43-48.

Lecordix, F., 2004. Research Service - *Innovative mapping techniques*. Copyright IGN Saint-Mandé, Service de la Recherche, Projet Carto2001, www.ign.fr.

Neuffer, D., Bell, M. and Woodsford, P. 2004. Cartographic Generalisation on the Basis of Model Generalisation. *Symposium Praktische Kartographie Königslutter 2004*, May 2004 and at: http://www.laser-scan.com/pdf/Cartographic_Generalisation.pdf

Ruas, A., 2000, Project AGENT: Overview and Results of a European R&D Project in Map Generalisation, *ICA Workshop*, November 2000, Barcelona, Spain.

Schürer, D., 2002. Ableitung von digitalen Landschaftsmodellen mit geringerem Strukturierungsgrad durch Modellgeneralisierung. PhD Thesis, *Insititut für Kartographie und Geoinformantion*, Universität Bonn.

West-Nielson, P., 2001. Generalisation, The Danish Experience. *Laser-Scan User Group*, Cambridge, June 2001.