CAPITALISATION PROBLEM IN RESEARCH – EXAMPLE OF A NEW PLATFORM FOR GENERALISATION: CARTAGEN

Jérémy Renard [1], **Julien Gaffuri** [1,2], **Cécile Duchêne** [1]

[1] *Laboratoire COGIT, Institut Géographique National*
Jeremy.Renard@ign.fr - Cecile.Duchene@ign.fr

[2] *Joint Research Centre, IES, SDI unit*
Julien.Gaffuri@jrc.ec.europa.eu

INTRODUCTION

Work capitalisation is an important issue to improve the usefulness of research and the link with production lines and commercial software, as it is essential to be able to reuse theoretical research results to perpetually enhance processes efficiency. Some discussions during past workshops have identified this particular point as a central question in research, especially for NMAs [Stoter 2005]. The problem of linking research and practice is also underlined by the recent EuroSDR project [Stoter & al. 2010]. Indeed, the creation of complete and shareable platforms to perform past and present research seems to be essential in a goal of capitalisation and efficient link with production lines. The aim of this paper is to focus on what should be necessary for such a platform in terms of system architecture, specific research needs and technical realisations.

For the particular case of COGIT laboratory, many pieces of research on geographical data generalisation have been held for twenty years, resulting in the development of many generalisation software components (algorithms, measures, models, etc…). Nevertheless, a global environment where all these components could be used altogether in a collaborative way is still missing: they run on different platforms and use different programming languages. To solve this issue, our work over the past two years focused on finding a way to capitalise all these research results by gathering these components in order to be able to launch a global generalisation process. A crucial part of this work is the creation of a research platform which gathers all previous works on generalisation and proposes new tools to make research more efficient, more powerful and more suited for sharing. That necessity led us to a think about the needs of such a software.

The aim of this paper is to begin a reflexion about research capitalisation through an innovating computer software. It first gives an overview of the general needs and objectives for a generalisation research platform, then focusing on the generalisation research case in COGIT laboratory. Then the chosen architecture covering the combination of different software and languages is presented, as well as the technical realisations that have already been achieved. The last section deals with the further needed improvements and the possible benefits for future research, both in and outside COGIT laboratory.

1. EXPRESSION OF NEED

In a context of generalisation research, there are key features that should be available in a research platform. The most important one is the possibility to share techniques between researchers and to externalize functionalities [Edwardes & al. 2007]. In an optic of pooling and capitalisation, that point is essential. In our own analysis, it covers the following issues:

- normalised coding and documentation to make collaboration easier, using the norm ISO 19107,

– possibility to integrate research work in other domains (simulation, legend conception, data integration, …),
– sharing of the interface without technical difficulties.

Moreover, building a software dedicated to a particular field of geographic sciences obviously raises the question of its fitness to the usual aspects of a Geographic Information System. The platform has to face efficiently the main functionalities of a GIS, especially those concerning data archiving, treatments and analysis, and geographical display. It is obvious that many different solutions already exist in the currently used generalisation systems, for instance:

– archiving : Gothic, ArcGIS, PostGIS, SHP files…
– analysis : generalisation models, Gothic treatments, java algorithms, …
– display: JADE/Clarity, GeOxygene, independent GUI, …

As capitalisation also throws the question of interoperability, all these software solutions should be performed one with another without any difficulty; that is a crucial point to build an efficient platform. A component-based software seems to be the most sensible approach, the requirements of such an architecture being adapted to our current needs [Rütschlin 2000]. Indeed, the very core of the platform should act as a hub which makes the different components work together, as illustrated in figure 1, using interfaces on which the components could reconnect. With such an architecture, it would be possible to disconnect one of the component and to replace it by another software solution, depending on the habits and on the needs of the user. So the platform could be easily shared by people using different work environments – for example from people using exclusively Gothic to people using Java algorithms on an independent GUI based on PostGIS. Based on this architecture, other components could obviously be added if necessary.

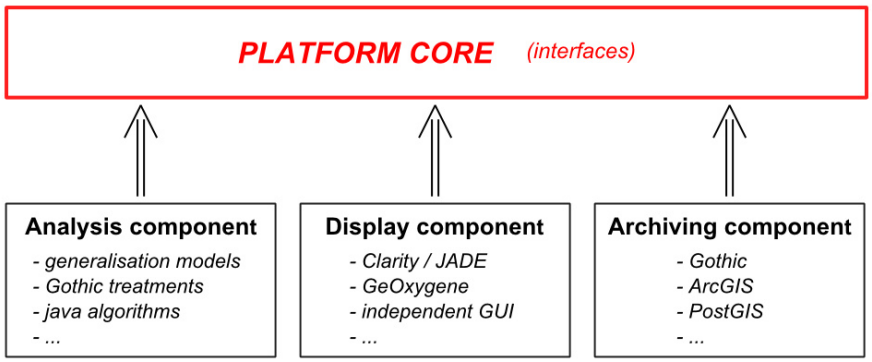

Figure 1 *- Theoretical component-based architecture for an efficient platform*

Some additional specific components are needed for research. For example, processes traceability tools are in order to control all what is done, to see the possible blocking points, and to improve the algorithms by correcting their core. Edition tools are also essential to create test areas and particular cases which might not exist in every dataset. Last, the possibility to load data from various sources seems to be a important point for the interoperability of the platform.

Besides, a particular need concerns the possibility to create generalisation web services in a context of on-demand mapping. The principles of such an evolution have already been raised [Neun & Burghardt 2005] [Regnauld 2007], and an important aspect of a generalisation platform is to be able to support the upcoming evolutions in this domain.

2. APPLICATION TO A SPECIFIC CASE: CARTAGEN

Several generalisation environments have been developed in the COGIT in the past twenty years, balancing between two different solutions: first platforms were totally dedicated to research (PlaGe and

Strategie), then commercial platforms (Lamps2 and Radius Clarity, from 1Spatial) were more suitable to link research results with production issues (see [Lecordix & al. 2005]). If these commercial platforms offer the great advantage of being easily sharable with production lines and to provide a helpful commercial support, they also throw additional constraints on the code control and evolution which led some pieces of research to be developed in other programming environments.

COGIT current generalisation research mainly relies on four models derived from PhDs results since 1999 (figure 2). Future works towards on-demand mapping should be based on these models :

- the AGENT model [Ruas 1999], developed on Lamps2 during the European project AGENT [Lamy & al. 1999] then transferred on the generalisation platform Clarity, and partially recoded on the COGIT platform GeOxygene [Badard & Braun 2003],
- the CartACom model [Duchêne 2004] developed on Lamps2,
- the GAEL model [Gaffuri 2008] developed on GeOxygene and partly transferred into Clarity,
- the knowledge revision component [Taillandier 2008] developed in Clarity and then transferred on GeOxygene.

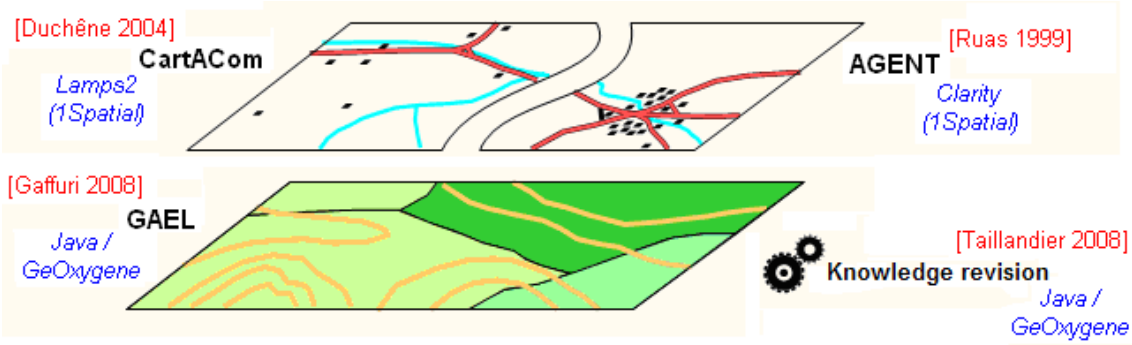

Figure 2 - *Generalisation models developed at COGIT, their most relevant application cases and their programming environment.*

The interactions between these models still raise modelling issues [Duchêne & Gaffuri 2008]. Apart from theoretical collaboration between the processes, the main technical problem is raised by the difficulty to compute together algorithms coded with different programming languages (Lull and Java) and running on different software (Lamps2, Clarity and GeOxygene). As a consequence, it appeared that a necessary step to improve our mastery on automated generalisation was to reengineer a programming environment which allows an integration of these different models in a complete generalisation platform. The first step of this capitalisation work is to create an orchestration model to make existing generalisation models – from COGIT and from other research – work together in a collaborative way [Touya & al. 2010]. In parallel, the second step is the development of this new generalisation platform called CartAGen (for CARTographic Agent GENeralisation), which is supposed to answer the needs developed in part 1.

These particular needs are not covered by our current research platform Clarity, whose goals are significantly different, so a new platform is necessary. A complete reengineering of the core of our models is needed, which cannot be done directly in Clarity since its core is not expected to be deeply modified. Being able to work easily on the core of the software is a necessary condition to make the different models collaborate. Nevertheless, there are some reasons why we cannot totally abandon it, especially because of its topology model which is very powerful. In addition of topology, many algorithms are working well in Clarity, and their recoding could be a huge waste of time.

There were other particular points which motivated us to change our work environment. The use of Clarity as a research platform also brings some additional inconvenience, for instance the lack of tools which could be useful for research (editing tools, process traceability, etc…). The aim of the new platform is also to provide these work tools for a better control on processes, and to create a more efficient work environment.

In fact, the main problem is to design a work environment which is suitable on the one hand for research process, and on the other hand for the link with production lines, but an ideal intermediate solution to face this paradox might not exist. In this optic, the adaptability of the platform to research rather than to production should be preferred.

All these specific needs led us to the absolute necessity to create a new computer software, and to the preliminary question of the architecture and the technical realisation of this generalisation platform in order to ensure a good capitalisation for past works.

3. SYSTEM ARCHITECTURE

It appears that the global answer to the need is partly split in two opposite solutions : using Clarity and its spatial DBMS Gothic, or developing in GeOxygene environment using another spatial DBMS like PostGIS. So it seems to be necessary to use both solutions and to make them work in collaboration, by connecting some GeOxygene libraries to Gothic treatments through Clarity. The question being: where to put the cursor between these two different aspects ?

The final goal of the platform is to propose a component-based structure: it is designed as a set of interacting software components, each of them focusing on a specific research problem and potentially using other lower level components. By this way, components can be easily reused, the platform can be extended and mature components could be transferred to production lines.

To achieve this goal, CartAGen is based on a first platform prototype developed to bypass some of the identified drawbacks of Clarity as a research platform [Gaffuri 2008]. This prototype was already using GeOxygene and Java code. Using this as a basis for further developments allows the addition of new tools and new functionalities without any difficulties. Indeed, the central problem is how to combine this external platform with a Gothic dataset to be able to use Clarity topology or special algorithms that are not recoded in Java.

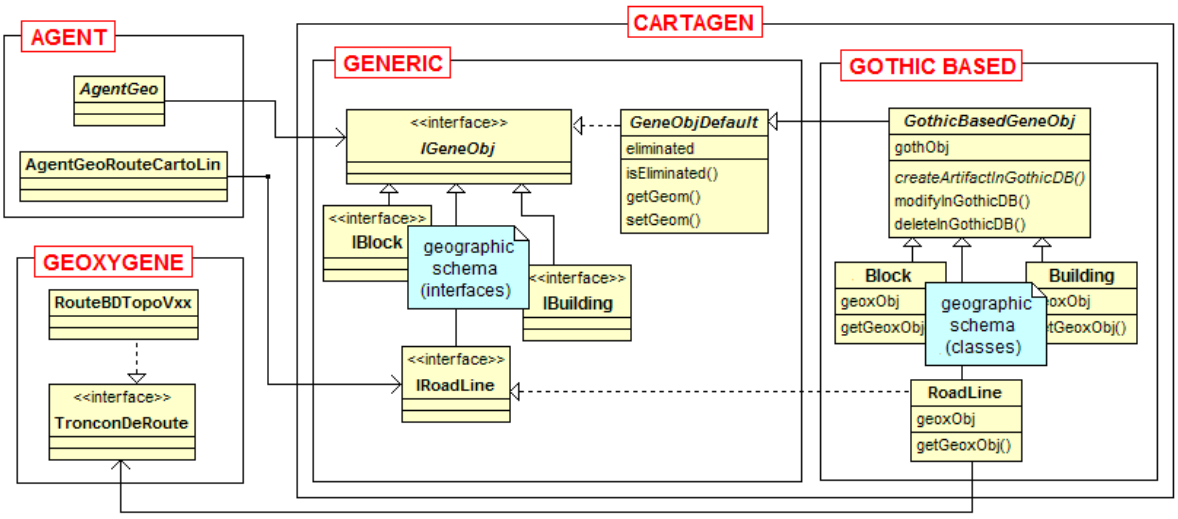

Figure 3 - *Data infrastructure to combine GeOxygene objects, agent process and Gothic environment. A complete geographic schema has been defined ; just a few classes are represented here.*

This sharp combination is based on a data structure that creates a Gothic dataset containing Gothic objects homologue to Java objects existing in CartAGen. As shown on figure 3, a generic geographic schema has been created, based on GeOxygene own schema. Each object of this schema is instantiated in a Gothic based part of the structure which ensures the link with the corresponding Gothic object in Clarity, and is delegated to a GeOxygene object. These two datasets (Java and Gothic) are updated in parallel, using one or the other depending on the treatment that is applied:

– topology treatments or special Clarity algorithms are computed on Gothic dataset,

– other treatments are computed on Java objects.

A significant advantage of such a structure is to clearly isolate all classes that rely on the Gothic part of the platform, so it could be possible to disconnect this Gothic based substructure if necessary ; it seems to be sensible to identify the DBMS – nowadays Gothic – as an own independent component of the platform. Indeed, in terms of modularity, we should consider the possible use of other spatial DBMS such as Oracle or PostGIS in order to ensure the platform portability. In this optic, that architecture allows to pass over the Gothic based part of the platform and to instantiate Java objects through an other DBMS or directly in the GeOxygene geographic schema. This separation makes the system management much easier and offers some relevant perspectives for further work.

It is significant to note that the theoretical component-based structure proposed in figure 1 is equivalent to the practical architecture presented in figure 3: external components are connected to the very core of the platform: generalisation treatments are ensured through the Agent model, and GeOxygene is used to display geographical objects in the GUI. Archiving component can be either Gothic or another GIS.

4. TECHNICAL REALISATION

The GUI illustrated on figure 4 totally relies on an original prototype [Gaffuri 2008]. The interface proposes most of the classical behaviours of GIS software, like layers management or data loading. Global generalisation can be performed using the agent process, but generalisation algorithms are also accessible independently and directly computable one-by-one through the different toolbars, so the user can have mastery on everything he wants to do – that point is a huge assess in a research process.

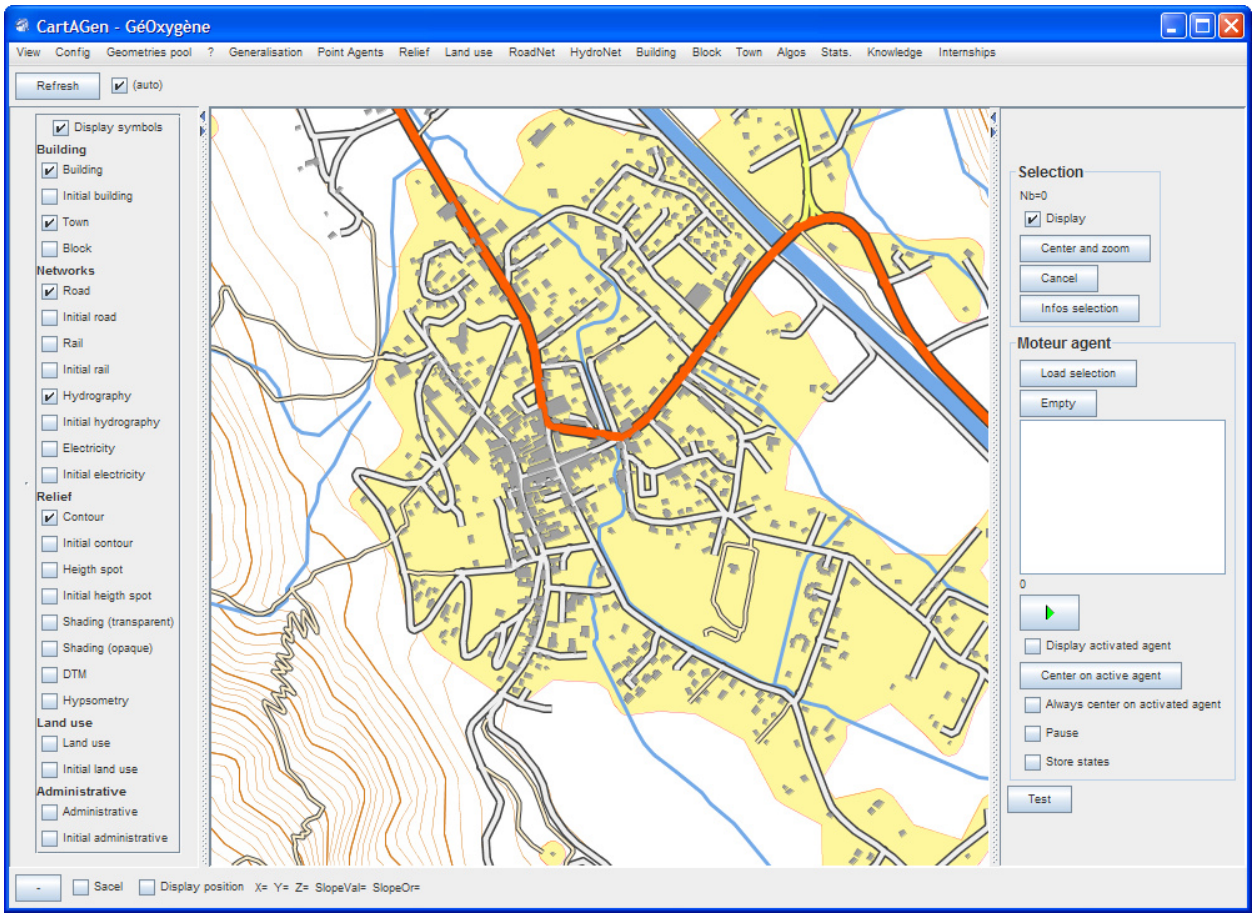

Figure 4 - *CartAGen GUI: layers are managed on the left panel, Agent process is launched using the right panel, and the top menu gives the access to independent algorithms and configuration properties*

The link between Java and Gothic is ensured by a cache mechanism managing the update of the Gothic dataset. Every created, modified or deleted Java object is put into the cache, which is committed to update the Gothic dataset every time a Gothic treatment is needed so the Gothic dataset is precisely corresponding to Java world every time it is used. This mechanism is a good way to save time, since the cache is never committed as long as no Gothic treatment is computed. Indeed, cache commit is a heavy step of computation because of the many geometry conversions it implies.

Many useful tools have been added to the first prototype, including for instance the possibility to add flying geometries over the data in order to test processes without any modification on the dataset. Some editing tools developed through an internship are about to be integrated to the interface. Figure 5 gives an example of another powerful research tool, which offers the possibility to replay any step of an agent generalisation process, displaying for each step the previous treatments that have been computed, the current agent satisfaction and a view of the current agent geographical state. Such a tool is very useful to perform traceability and to control every step of a process, and gives interesting perspectives to detect and improve the incorrect parts of a process.

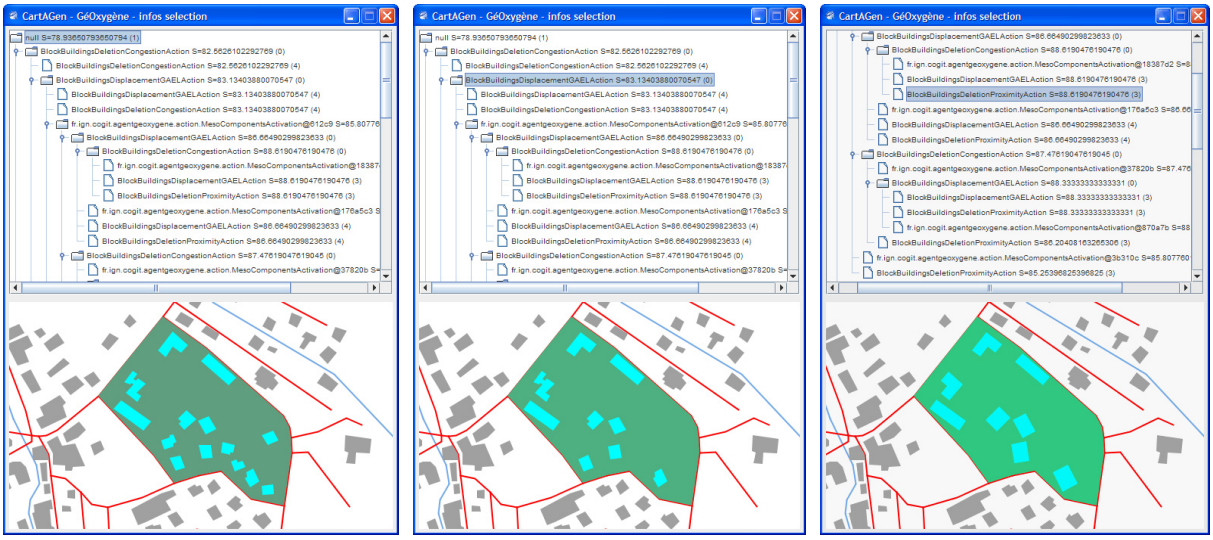

Figure 5 - *Generalisation tree, allowing to replay an Agent generalisation process.*

Through our research work, CartAGen has also been tested with significant differences in the components used. For instance, instantiating generic CartAGen objects through Gothic instead of GeOxygene and using Clarity as a display component works well and offers new possibilities in the use of the platform. Concerning analysis and treatments, the co-existence of Agent and CartACom models doesn't raise any problem. Archiving through SHP files instead of Gothic datasets is in progress and will be soon fully operational. These few tests prove that the component-based architecture is sensible and that it allows large possibility in terms of interoperability and sharing.

5. FURTHER WORK AND PERSPECTIVES

Most of the work has already been done concerning the core of the platform. Particularly, the link between GeOxygene (Java) and Clarity (Gothic) with a cache mechanism has been coded and seems to be sensible. The only thing that is still remaining on this particular point concerns topology creation and management through the cache. AGENT and GAEL models are completely integrated to the platform, CartACom recoding in Java is in progress. The integration of the module for knowledge revision will be performed later. The current work mainly consists in recoding in Java – or encapsulating Lull code – the algorithms developed in the past PhDs and studies, in order to gradually gather all developments at the same place and to be able eventually to compute global generalisation on large datasets.

With the capitalisation of all processes and models, and the addition of powerful tools to help research, the final goal of the platform is to make our research work faster, easier and more efficient. Even if the

platform realisation is not fully achieved, its first uses allow us to already notice some of its interesting perspectives. For instance, it should be a well suited environment to move towards the development of generalisation web services in a context of on-demand mapping. Overall, the interoperability ensured by the component-based architecture offers new possibilities to share models and algorithms with external researchers and is a real promise for future collaborative work.

REFERENCES

Badard T., Braun A. (2003), *A distributed system architecture to provide on-demand mapping*, proceedings of the 20th International Cartographic Conference (ICC'03), 6-10 August 2003, Beijing (China), 994-1004

Duchêne C., Gaffuri J. (2008), *Combining three multi-agent based generalisation models: AGENT, CartACom and GAEL*, proceedings of the 13th International Symposium on Spatial Data Handling (SDH'08), 23-25 June 2008, Montpellier (France), 277-296

Duchêne C. (2004), *Généralisation par agents communicants : le modèle CARTACOM - Application aux données topographiques en zone rurale*, PhD thesis in computer sciences, University Paris VI - Pierre et Marie Curie, 11 June 2004

Edwardes A., Burghardt D., Neun M. (2007), *Experiments in building an open generalisation system*, in Ruas A., Mackaness W., Sarjakoski T., *Generalisation of Cartographic Information: Cartographic Modelling and Application*, 161-175

Gaffuri J. (2008), *Généralisation automatique pour la prise en compte de thèmes champs: le modèle GAEL*, PhD thesis in computer sciences, University Paris-Est Marne-la-Vallée, 1 July 2008

Lamy, S., Ruas, A., Demazeau, Y., Jackson, M., Mackaness, W. A., Weibel, R. (1999), The Application of Agents in Automated Map Generalisation, proceedings of the 19th International Cartographic Conference (ICC'99), 14-21 August 1999, Ottawa (Canada), 1225-1234

Lecordix F., Jahard Y., Lemarié C., Hauboin E. (2005), *The end of Carto2001 Project TOP100 based on BD Carto® database*, 8th ICA Workshop on generalisation and multiple representation, 7-8 July 2005, A Coruña (Spain)

Neun M., Burghardt D. (2005), *Web services for an open generalisation research platform*, 8th ICA Workshop on generalisation and multiple representation, 7-8 July 2005, A Coruña (Spain)

Regnauld N. (2007), *A distributed system architecture to provide on-demand mapping*, proceedings of the 23th International Cartographic Conference (ICC'07), 4-10 August 2007, Moscow (Russia),

Ruas A. (1999), *Modèle de généralisation de données géographiques à base de contraintes et d'autonomie*, PhD thesis in computer sciences, University Marne-la-Vallée, 9 April 1999

Rütschlin J. (2000), *The requirements for a component-based architecture*, proceedings of the ProSTEP Science Days 2000 "SMART Engineering", 13-14 September 2000, Stuttgart (Germany), 252-260

Stoter J. (2005), *Generalisation: the gap between research and practice*, 8th ICA Workshop on generalisation and multiple representation, 7-8 July 2005, A Coruña (Spain)

Stoter J., Baella B., Block C., Burghardt D., Duchêne C., Pla M., Regnauld N., Touya G. (2010), *State-of-the-art of automated generalisation in commercial software*, EuroSDR project report

Taillandier P. (2008), *Révision automatique des connaissances guidant l'exploration informée d'arbres d'états - Application au contexte de la généralisation de données géographiques*, PhD thesis in computer sciences, University Paris-Est Marne-la-Vallée, 2 December 2008

Touya G., Duchêne C., Ruas A. (2010), *Collaborative generalisation: Formalisation of generalisation knowledge to orchestrate different cartographic generalisation processes*, 6th international conference on Geographic Information Science (GIScience'10), 14-17 September 2010, Zurich (Switzerland), accepted as full paper