Chapter 7

Process modelling, web services and geoprocessing

N. Regnauld, G. Touya, N. Gould, T. Foerster

* Author 1 coordinates
** Author 2 coordinates
*** Author 3 coordinates

Abstract.

Process modelling has always been an important part the research in generalisation. While in the early days, this would take the form of a static sequence of generalisation actions, today the focus is on modelling much more complex processes, capable of generalising geographic data in to various maps according to specific user requirements. This chapter discusses several aspects of the problem of building such systems. As the system get more complex, it becomes important to be able to reuse components which already exist. Web services have been used to encapsulate generalisation processes in a way that maximises their interoperability and therefore reusability. However, for a system to discover and trigger such service, it needs to be described in machine understandable way, and the system needs to have the knowledge about where and when to use such tool. This chapter therefore explores the requirements and potential approaches to design and build such systems.

Keywords. Keyword 1, Keyword 2, Keyword 3

1. Introduction

1.1. State of the art

Early research on automated generalisation focused on developing algorithms to automate single generalisation operations (Jenks 1989, Douglas & Peuker 1973, Jäger 1991). After a few of these emerged, the next logical question was how to combine them to generalise an entire map? This is why the research community started to focus on process modelling. The first studies proposed static sequences of algorithms to derive a specific map, this is often referred as batch processing. (Weibel et al 2007) explain how this domain has evolved into more advanced types of modelling, which the authors have classified into three main categories: condition-action modelling (rule based system), human interaction modelling, and constraint-based modelling. The rule based system, which has sometimes been implemented as an expert system, was very popular at first, but proved difficult to extend beyond a certain point, due to the difficulty of formalising and capturing all the rules required, and to avoid inconsistencies when the number of rules grows. While inconsistencies in a rule set can be tracked down (Brenner and Sester 2005), resolving them can be very difficult. The human interaction model was letting the human make most decision of what tool to apply where, but such system offer a limited increase in productivity. In order to overcome the limitations of these two models, the constraint-based model was developed. The principle is to express the requirements for the target map in terms of constraints (see Chap. 2), and to use a mechanism to trigger a combination of algorithms to maximise the satisfaction of this set of constraints. Different models have been studied to reach the optimum:

- Multi agent model
- Combinatorial optimisation model
- Continuous optimisation model.

Since the review of (Weibel et al 2007), no ground breaking advances have been made in this domain. Instead, the existing techniques have been refined, and applied to real world use cases with very positive results. The most interesting advance on the methodology front has been to add another layer to the decision process, and apply different subsystems to different tasks, or different geographic configurations (see Sect 7.2).

Chap. 11 provides an overview of the most recent success in applying automated generalisation to real production systems, all of them implemented in National Mapping Agencies, using different technologies, and achieving different level of automation on different types of products. Systems described in Chap. 11.3, Chap. 11.4 and Chap. 11.8 produce paper maps at scales between 1:25k and 1:100k from topographic data, respectively at IGN France, at SwissTopo and in a group of German federal states (Bundesländer). In these three cases a first pass applies automatic generalisation to the data, before manual editing takes place to finish the maps. IGN uses Radius Clarity (1Spatial) for the automated generalisation followed by GeoConcept Publisher for the manual editing. SwissTopo uses Axpand (Axis Systems) in combination with ArcGIS (ESRI). The German solution is also based on Radius Clarity for the automated generalisation part. Chap. 11.5 describes another system using automated generalisation, but this time the generalisation system is fully automated, and manual finishing is not required. The product obtained is not the usual high quality topographic map, but a lighter backdrop map, designed to be used at scales around 1:25k for overlaying other data onto it. The generalisation system is based on 1Spatial software (Radius Clarity and Radius Studio). Chap. 6 and Chap.7 present automated generalisation processes developed respectively at USGS and Kadaster NL using ArcGIS (ESRI). All these systems have been built by heavily customising the original platform. In some cases the customisation and extensions have been made by the software company itself (1Spatial for the German federal states, Axis Systems with SwissTopo), sometimes by the customer (IGN and OS). So platforms exist for developing complex automated generalisation processes that bring real benefits to production systems. What we have not seen yet is a platform that can be used out of the box by simple configuration (importing the input data into the system and expressing the requirements). This is one of the main objectives of the current research on on-demand mapping. The general idea is to build a system that can easily integrate data from different sources, and generalise them together to obtain a map as specified by the user. The general idea has been described in (Regnauld 2007), and refined since (Foerster et al 2012). (Balley and Regnauld 2012) proposes an architecture for such a system which decouples its main components. These are summarised in Figure 1. They include

- The product specifications, to formalise the requirements
- The data access manager, to link the data with the system internal schema
- Web services, to provide algorithms (generalisation algorithms, spatial analysis tools, etc.), which should be formally described

- The knowledge base, which contains all the knowledge required by the system (procedural knowledge, cartographic knowledge, geographic knowledge) (Armstrong 1991).
- The engine, which is capable of using the knowledge to interpret the specifications and build and apply the workflow that will derive the required map from the given data, using the tools available.

Such a system, in order to deliver true on demand mapping, would be very

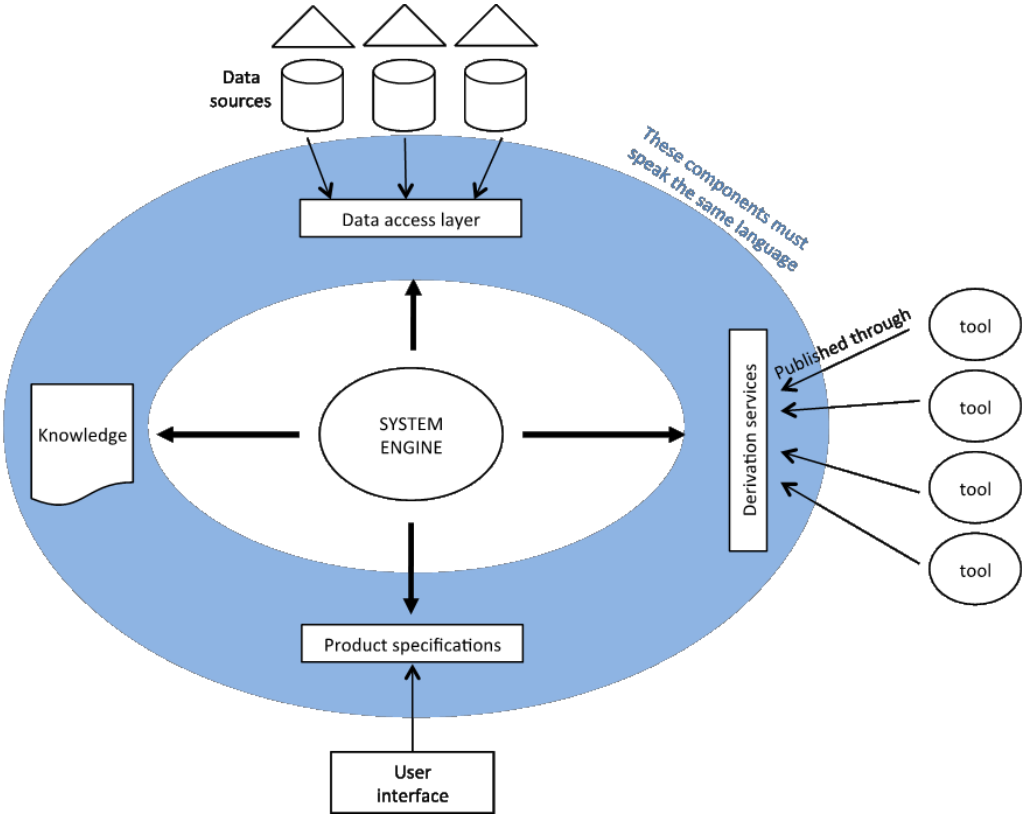


Figure 1: High-level architecture deriving on-demand products

big and complex, including large libraries of well described tools and rich knowledge of all kinds. Building such a system should be done incrementally, focusing on one of few types of requirement at a time, and enriching the system with the components required. This brings us to discuss the componentisation of the generalisation process.

1.2. Componentisation of the generalisation process

Componentisation is the process of decomposing a complex system into simpler component. In our context we look into decomposing a generalisation system into components that can be developed independently and shared. These components can be simple algorithms, or much more complex tasks. The benefit of having complex tasks is that they can encapsulate the knowledge specific to the task. This can reduce the amount of knowledge required in the top level system. This is a way of organising the knowledge to avoid having a huge flat set of rules and constraints which would quickly become unmanageable and impossible to extend. Figure 2 shows an example of complex task from (Balley et al 2012).

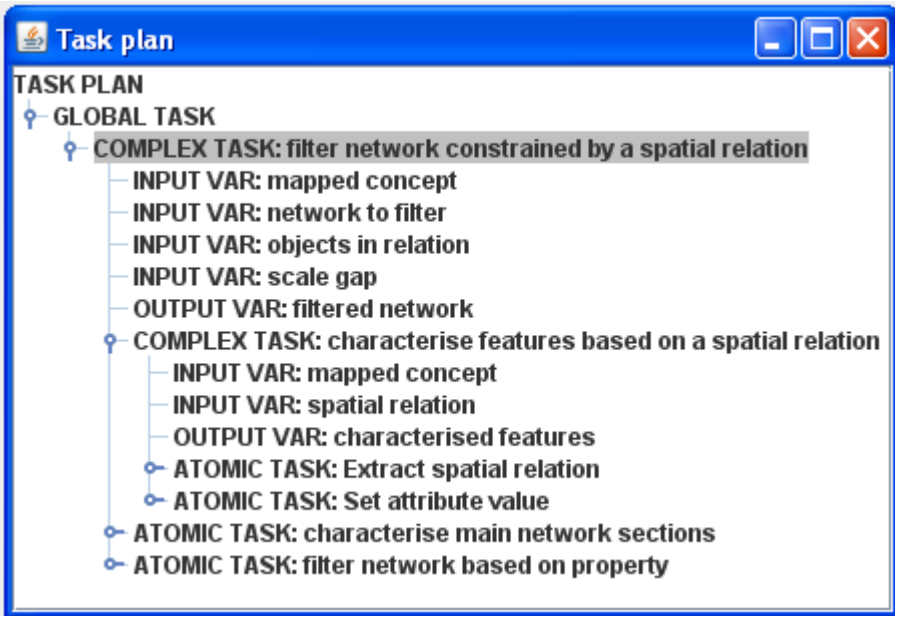


Figure 2: Example of a complex task

This task was instantiated for pruning a road network (*network to filter*) while making sure that roads supporting (*spatial relation* alignment) cycle routes (*mapped concept*) were kept. This task is made of three subtasks. The first one is a complex task in charge of marking the sections of the road network which are aligned with a cycle route. The second task is atomic and identifies the main roads in the road network. The final atomic subtask performs the filtering based on the attributes set during the two previous subtasks.

This is also a good way of reusing existing processes to incorporate them into bigger, more complete systems. For example Touya & Duchêne (2011) combines existing processes to make a generalisation system capable of handling different complex situations. Balley and Regnauld (2012) have proposed a model using tasks and subtasks for organising the knowledge in their on-demand mapping system. Goals are chosen depending on the user requirements, and then each goal is associated with a task designed to achieve it. Each task encapsulates its own procedural knowledge, used to trigger the appropriate actions.

To ensure interoperability between these tasks, there is a need for them to conform to some standard. The requirements for a shared development platform for generalisation has been identifies by Edwardes et al (2003). It had proposed a number of possible approaches, and led to the development of WebGen, a client-server platform for sharing processes using Web services (Neun and Burghart 2005). Here we review the different ways of sharing processes:

- Open source. There are a number of open source projects (Open JUMP, GeoTools, QGIS, GeOxygene, 52°North WPS, GRASS, sextante) that propose a platform for developing geospatial processes. Users can either develop on the platform, or import some libraries into their own platform to use existing tools. This approach has limitations though, as the integration can be cumbersome (incompatible open source license models, linking libraries and writing translators to cope with different data models), and needs to be done every time a new library is required. There are also potential compatibility issues. Users are often already using their own development platform, integrated with their own systems. Switching to a new one is not often possible. However, using libraries of generic tools is a common way of reusing existing open source software (JTS, JCS, CGAL, etc.).
- Web Services. The Web service approach is a way to package processes in a standard way. Web services are hosted on servers and can be accessed by remote clients through the web. The main advantage of the web service is that the client does not need to run on the same platform as the service it is calling. This provides true platform interoperability. It is also very versatile, as there is no limit to what can be encapsulated in a service. It could be a simple generalisation operator, or even a simple measure, or it could be a full generalisation system. Web services for generalisation have been studied. An initial platform called WebGen was developed at the University of Zurich (Burghardt et al 2005, Neun and Burghardt 2005). It was later adapted to the OGC standard WPS, at the

request of several members of the ICA Commission on Generalisation and Multiple Representation (Foerster et al 2008). Tests have been done at Ordnance Survey to demonstrate the cross platform interoperability benefits of the approach. A server capable of hosting services that rely on the platform Radius Clarity has been setup. These services have been successfully called by an OpenJump (http://openjump.org/) client. A client for ArcGIS is currently under development at 52°North, funded by the Ordnance Survey (GB). More information and updates on WebGen-WPS can be found on the commission website (http://generalisation.icaci.org/index.php/web-services).

- Code moving. A disadvantage of the web service approach is that it involves moving the data from the client to the server, doing the processing remotely and downloading the results; this can involve transferring large amounts of data. An alternative approach would be to send the executable code to the client, where it can execute the process locally. This code moving concept and its application to geoprocessing is described by Müller et al (2010, 2012). It has the advantage over the web services to require less data transfer, as the size of the code is often much smaller that the size of the geographic data being processed. This could therefore result in much faster overall execution time. It also reduces the risks related to data security, as the data does not leave the client. The main problem with the method is that it requires the client to provide a compatible runtime environment, compatible hardware, and there may also be licencing issues if the code includes third party libraries. This approach also allows organisations to publish their tools without having to maintain a powerful server capable of running the process locally. This technology is currently less mature, less readily available than the web service approach, but the concept is interesting.

Both the web service and the code moving approaches require a formal description of the process proposed. This includes a high level description of what the service does, and also the type of all the parameters required. This is often referred as the service contract, which formalises the requirement of the service to ensure its successful execution. This should be enough for a human to choose what service to use. In a context of automatic discovery of services, it becomes essential that these descriptions are formalised and use a standard vocabulary. This has been discussed in (Balley and Regnauld 2011) and led to the definition of the *semantic referential*, which defines all the concepts that need to be shared by all the components of their on demand mapping system. (Touya et al 2010) also identified similar requirements, and propose a generalisation domain ontology to define these concepts. Geographic concepts relate to real world objects and can be easily defined (see for example those defined by INSPIRE), even if reaching a con-

sensus is often difficult. Many classifications of generalisation operators have also already been proposed (McMaster & Shea 1992, Regnauld & McMaster 2007, Foerster et al 2007), so again, the challenge is to adopt a common one. It becomes more difficult when it comes to formalising the description of the parameters, as these are sometimes very specific to a particular process (cf. figure 2 in 2.1.2). The best way to overcome this seems to be to rely on translators that derive the values of the parameters required by a service from formalised user requirements.

1.3. Formalising the procedural knowledge

Procedural knowledge is used to guide the selection and application of generalisation operators (Armstrong 1991). These can take various forms like rules, constraints or ontologies. Technically this is declarative knowledge, but we often associate this to the domain savoir-faire, which we include in the procedural knowledge. This selection mechanism depends on the input data, the output required and of course the operators available. Formalising this knowledge therefore requires an existing formalism to describe the types of geographic features handled, the types of operators required, and the requirements, often referred as map specifications (chapter 2).

The procedural knowledge can take many forms. In its "unformalised form", it can be found in the implementation of batch processes, as a static sequence of operations, possibly enhanced by the use of conditional statements to adapt the sequence to the conditions. In more advances system, like those based on optimisation techniques, the procedural knowledge provides the heuristics used by the system to explore the space of solutions. This is rarely well formalised. Taillandier (2011, 2012) formalises part of the procedural knowledge in the AGENT system in order to revise this knowledge automatically, to improve its performance.

Formalising the procedural knowledge is part of formalising all the knowledge required by the system. Concepts used to describe the procedural knowledge must match those used to describe the map specifications (chapter 2).

The key components of the system that needs to be formally described are the basic tools (generalisation operators, measures, spatial structures, etc.). Some elements of the description are purely functional, while others are dependant on how that are made available (web service, moving code, library).

Items required for describing tools include:

- Type of operation performed (see chapter 7.3)

- Type of data processed. Many data models for MRDB have been developed in the past, for example MADS(Parent et al 1998) proposes a hierarchy of spatial abstract types that can be used to categorise the geographic classes. Ontologies have also been used to organise and describe geographic data and their interactions see chapter 2.3)
- Geographic context (scale, type of geographic area, etc.)
- Parameters
- Software dependencies (for code moving)
- Hardware dependencies (for code moving)

Knowledge is also required to link the tools to the conditions in which they can be used. These conditions are influenced by various factors, some coming from the requirements (target map specifications), others from local context. Several models have been defined to try to automatically choose the tools based on the requirements and context:

- Model developed by Balley (2012) (goals and tasks)
- Model developed by Touya (see chapter 7.2)

1.4. Orchestration

Once a library of tools to analyse the data and transform them is available, that the specifications of the products are available, the next challenge is design and build the workflow that will trigger the appropriate actions in the right order.

This part will discuss the different ways processes (available as services or otherwise) can get chained together.

- Workflows. Using workflows to link processes is the usual way of chaining existing processes. These are in general very static, built on top of the existing tools (Burghardt et al 2010).
- Service chaining is the process of building a sequence of individual services to perform a more complex task. Depending on how the services are described, this chaining can be done in different ways. When the services are not or poorly described, the chaining has to be done by an expert who knows exactly what task the service is performing. For services well described using natural language, the service can be used by someone who needs not know how the service works, but understands what it delivers. When the service description is formalised, it opens the door for

chaining being automatically done. A chain of services can be encapsulated as a new service. Services can therefore be made available for all sorts of tasks, from the atomic operations to the full generalisation process.

Standards like BPEL (Business Process Execution Language) to assemble services have been developed. (Schaeffer & Foerster 2007) present an approach that uses it for chaining OGC services.

Service chaining is to some extent similar to creating a workflow using services. However it adds the interoperability aspect, and as service get described with more formal languages, the chaining should soon include aspects of automatic service discovery which provides a much more dynamic way of chaining services.

- AGENT System: In the AGENT system (Ruas and Duchene 2007), constraints are used to guide the choice of actions. Each constraint proposes a list of actions that could be triggered to attempt to increase the satisfaction. The agent system has an optimisation engine that looks for an optimum global satisfaction of all the constraints. While the current implementations of AGENT use actions available on the same platform, they could easily be replaced by service calls. The interesting aspect of the system is that the chaining of actions is built dynamically.
- Hybrids. A hybrid system combines different approaches to chain processes or services. These systems are usually designed using a very pragmatic approach, trying to reuse what is already available and combine them. For example, CollaGen (chap 7.2) uses a workflow to control different generalisation subsystem in charge of generalising a specific type of geographic context. These subsystems are already complex systems, some based on the AGENT paradigm described above. The model proposed by (Balley and Regnauld 2012) uses goals and tasks to dynamically build the sequence of high level tasks to derive the map specified by the user. Each task contains its own knowledge base that allows it to build its own chain of subprocesses (in this case implemented as services).

1.5. Opportunities opening up

- Cloud computing

With the current trend to componentise the process of generalisation, and allow interoperability between different platforms, exploiting the power

offered by parallel processing is becoming a very realistic prospect. Frameworks like MapReduce (Dean and Ghemawat 2004) or Hadoop (http://hadoop.apache.org/) have been designed for processing large datasets efficiently. They rely on mechanisms to split the task and distribute the processing over a cluster of processing nodes. While this is already widely used in other application domains, we still haven't seen it used for generalisation. However, we can easily imagine that once the tasks of a workflow have been identified as independent, they could be computed simultaneously in the cloud. Computing in the cloud also offers the benefit to user of getting the processing power they need when they need it, without having to maintain expensive hardware and software in house.

- The other benefit that componentisation can offer will require these components to be described in a standard machine readable way. Once this happen, there will be no need to put into workflows in into the procedural knowledge any direct reference to specific algorithm implementation (or services). The procedural knowledge will only mention the abstract operations required. Then depending on circumstances (input data, geographic context, requirements), the right service could be dynamically chosen. Of course, unless everything is perfect, there will be a high degree of failure (algorithm not stable, description misleading, lack of requirements). This is where optimisation systems such as AGENT can be used. They would be able to select several potential candidates, try them and keep the best one. This of course will add a considerable overhead, but this can be mitigated in two ways. First trying alternative solutions can be done in parallel. In addition, we can use automatic learning to record past experiences and use it to refine the knowledge used by the optimisation engine, so that it gradually stops trying options that tend to never deliver good results, and try those which are likely to provide satisfying results first. These self-optimising techniques have already been studied for an agent system performing generalisation, and proved very successful (Taillandier et al 2011)(Taillandier and Gaffuri 2012). With more distributed systems on the horizon, more optimization of this type will be required to efficiently harvest the power of Web based processing.

2. Case Study I: Collaborative Generalisation (by Guillaume Touya)

2.1. Principles of Collaborative Generalisation

Past and current research shows that existing automatic generalisation processes are not able to correctly generalise a complete topographic map, despite some very good results on some specific parts (*i.e.* landscape like urban or rural, or theme like roads or land use) of the map (Touya 2008). As a consequence, generalising a complete map requires the optimal use of the available (on a software platform or via web services) processes.

1.1.1. Collaboration of Automatic Generalisation Processes

According to McMaster and Shea (1988), an automatic generalisation process has to be able to know how, where and when to apply a generalisation operation. Collaborative Generalisation (Touya et al. 2010) aims at answering the same questions but at the upper level of processes: processes collaborate to apply on the part of space they are best suited for. Fig. 1 shows a schematic view of collaborative generalisation principles. The map is partitioned into spaces representing a landscape (*e.g.* urban, rural areas) or a data theme (*e.g.* the road network) (Touya 2010). Each of the spaces that needs generalisation is matched to the best suited available process, then the potential side effects at the space boundary are corrected.

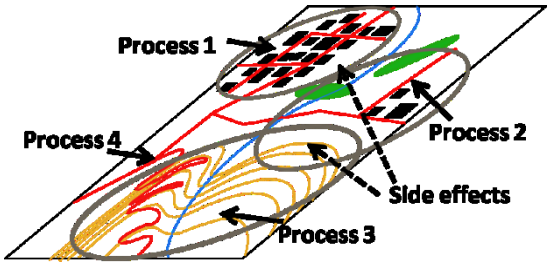


Fig. 1 The collaboration principle between generalisation processes: process 1 is carried out on the town area, process 2 on the rural area, then process 3 on the mountain area and finally process 4 on the road network. Side effects are corrected at spaces boundaries

1.1.2. Interoperability Problems with Collaborative Generalisation

The collaborative generalisation mechanism intends to make processes collaborate whereas they were not designed for and some interoperability issues are raised. Interoperability issues derive from different kind of heterogeneities in the collaborative generalisations (see sections 1.2):

- *Process capabilities description*: the capabilities of generalisation processes have to be specifically formalised to be able to choose the best suited one for a given part of space.
- *Parameter heterogeneity*: each process is monitored by its own set of parameters while the overall generalisation should have a single way of parameterisation (Fig. 2).
- *Evaluation heterogeneity*: collaborative generalisation is an iterative process that requires a generic self-evaluation model, independent from the processes.
- *Global syntactic and semantic heterogeneity*: process capabilities, parameters or evaluation rely on shared generalisation knowledge to avoid syntactic and semantic heterogeneity.

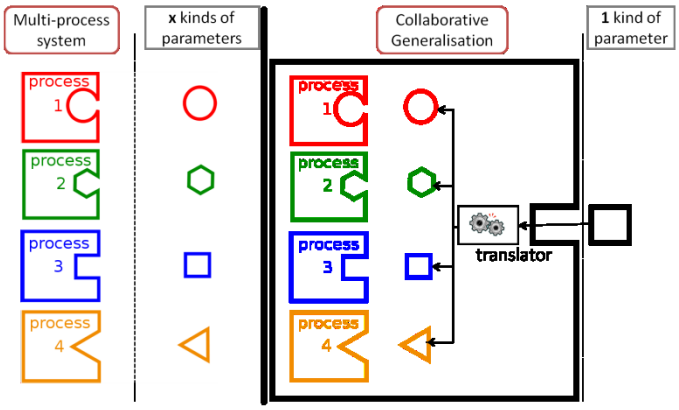


Fig. 2 Parameter heterogeneity requires a standard kind of parameter and a translator

2.2. Knowledge Formalisation for Collaborative Generalisation

The CollaGen model (Touya & Duchêne 2011) allows carrying out collaborative generalisation. Thus, some CollaGen components were developed to overcome the interoperability problems raised in the previous section. This section briefly describes each of these components with some experimental results.

1.1.3. Generalisation Knowledge Ontology

In order to solve syntactic and semantic heterogeneity, CollaGen is fitted with a generalisation knowledge ontology (see section 1.2). It contains shared vocabulary on geographic entities (*e.g.* building), and generalisation concepts (*e.g.* meso object) and operations (*e.g.* typification), or spatial relations (*e.g.* rivers flow into talwegs) (Touya et al. 2010). CollaGen formalised knowledge presented in the next section shares vocabulary by always referring to this ontology.

1.1.4. Specifications Formalisation by Constraints and Rules

The chosen standard format for specifications in CollaGen is a formal model for generalisation constraints (see Chapter 2). The formal model is described in Fig. 8 of Chapter 3: a generalisation is on a geographic entity (*e.g.* building), about a character (*e.g.* area) with a type of expression (*e.g.* character value < threshold).

Stoter et al. (2009) noticed that, sometimes, specifying the operation to do or not do was useful (*e.g.* small buildings shouldn't be aggregated), so we added a formal model for such rules (Touya et al. 2010). 80 formal constraints or rules have been defined in CollaGen prototype.

Then, a translator function is associated to each available process to transform the formal constraints and rules into the specific parameters of the process (Touya et al. 2010).

1.1.5. Constraint Monitors for Interoperable Evaluation

The CollaGen model needs to check the satisfaction of the formal constraints in the map during its iterative process, in a process-independent way, so constraints *monitors* are created in the data to monitor each formal constraint for each object concerned by the constraint (Touya & Duchêne 2011). Constraints monitors allow knowing where the map is badly generalised (Fig. 3). Generalising a complete map implies the handling of a huge number of monitors and evaluate their global satisfaction is a challenge (Touya 2012).

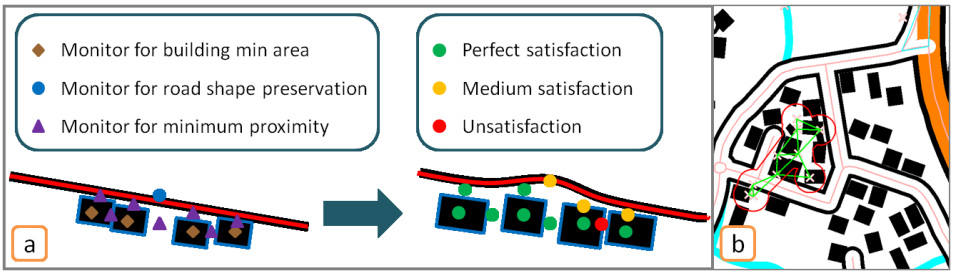


Fig. 3 (a) Constraints monitors for generic evaluation of generalisation (b) a group of unsatisfied monitors identified by CollaGen prototype

1.1.6. Generalisation Process Capabilities Description

Formalising the description of capabilities is a key issue in the quest for web services interoperability. Following the ideas of Lutz (2007) for geoservices, CollaGen describes generalisation processes with pre-conditions (*i.e.* the conditions the input data have to meet to be properly processed) and post-conditions (*i.e.* the expected data modifications caused by the process). For generalisation processes, pre-conditions are the type of spaces

the process is able to treat (*e.g.* CartACom process (Ruas & Duchêne 2007) is able to treat rural spaces) and post-conditions are the constraints that are expected to be satisfied after generalisation (Touya et al. 2010). In addition to the conditions, the formal description contains an optimal scale range and a list of required enrichments (Mackaness & Edwards 2002) (*e.g.* AGENT (Ruas & Duchêne 2007) requires blocks). In Fig. 4, AGENT gives results worse than CartACom on a rural area which can be specified in the formal description: AGENT's pre-conditions contain 'rural space' with a medium confidence ratio (3/5) and its post-conditions do not contain the 'building/road parallelism preservation' constraint, as opposed to CartA-Com.

1.1.7. Formalising Orchestration Knowledge

Finally, in order to orchestrate the generalisation order of the different part of spaces, CollaGen allows the definition of orchestrating rules that play Ruas & Plazanet's (1996) *Global Master Plan* role. It is able to express that 'the road network should be generalised before the rural areas', in a rule format (Touya et al. 2010). In Fig. 4, result (4), where the road network has been generalised before the rural area, is better than result (3), showing the value in specifying such rules.

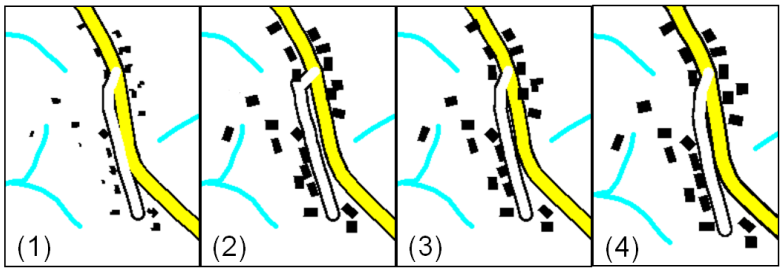


Fig. 4 (1) before generalisation. (2) generalisation with AGENT then Least Squares process (Harrie & Sarjakoski 2002) (3) CartACom then the Beams (Bader et al 2005) (4) the Beams then CartACom

2.3. CollaGen results

The CollaGen model is based on the formalised knowledge to carry out automatic collaborative generalisations (Touya & Duchêne 2011). A prototype was developed with access to nine automatic processes, all dedicated to topographic maps. Fig. 5 shows results for a 1:50k map that contains urban and rural landscapes. These results have been evaluated as much better than any automatic process used alone. Fig. 6 shows that CollaGen performs even better than the best (non automatic) benchmark tests with commercial software from the EuroSDR tests (Stoter et al. 2009), thanks to the collaborative generalisation principles.

Fig. 5 Extract of a CollaGen generalisation on a large area at the 1:50k scale, where nine processes collaborated

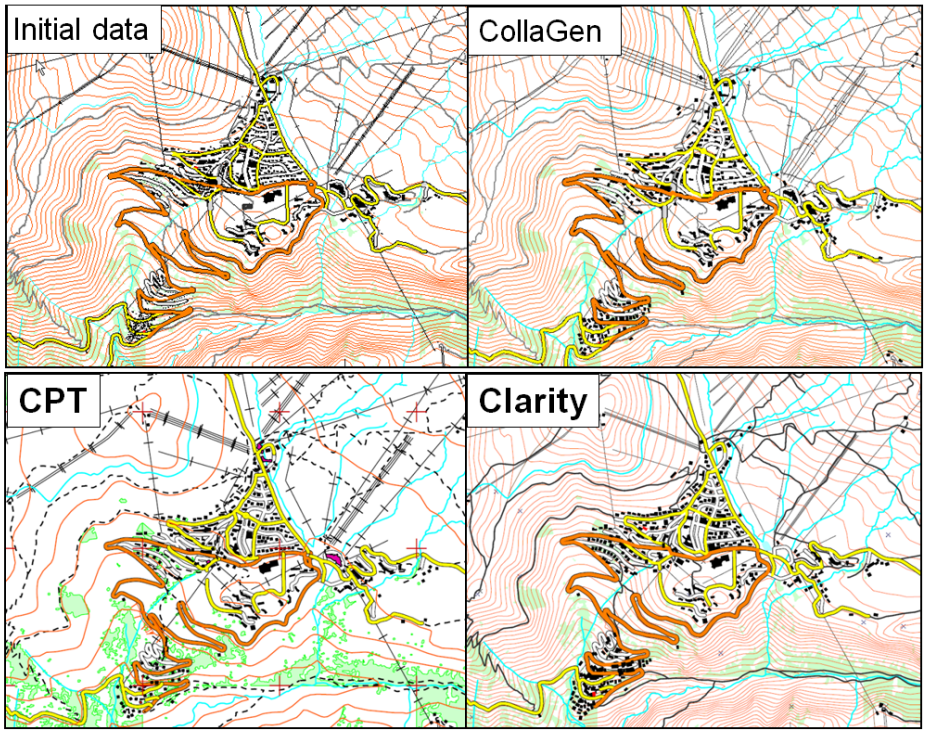


Fig. 6 A mountainous French dataset from EuroSDR tests (Stoter et al. 2009) generalised with CollaGen compared to the best results from the tests

3. Gould – An Ontological approach to On-demand Mapping – Generalisation

3.1. The case for ontology-driven generalisation

This study relates to on-demand mapping and in particular focusses on knowledge formalisation and how it can be used to aid the automatic selection of generalisation operators and algorithms. If we wish to automate any process then we must *formalise* the knowledge for that particular domain. The knowledge needs to be machine understandable not merely machine readable. In that way the system can make decisions based on the knowledge it has of the process. The acquisition and formalisation of cartographic generalisation knowledge has not proved easy (Rieger and Coulson, 1993; Kilpeläinen, 2000). Consider, for example, the naming and classification of generalisation operators.

As discussed (section 1.2), there have been numerous attempts to classify and describe generalisation operators but the problems highlighted by Rieger and Coulson (1993) remain. As well as differences between the proposed categories of operators there are also problems when different terms are used for the same concept (Aggregation or Combine?) and in granularity; McMaster and Shea (1992) define Smoothing, Enhancement and Exaggeration where Foerster et al. (2007) simply define Enhancement. There is also disagreement as to what functions can be regarded as generalisation operators. For example, is Symbolisation a generalisation operator (McMaster and Shea, 1992) or a pre-processing step (Foerster et al., 2007)?

The use of different operator taxonomies in closed systems does not matter, but, if we are to develop an interoperable on-demand system, an agreed taxonomy as well as the semantic description of the operators is required. This is because we cannot simply ask for a web service that performs Smoothing, say, since that operation can be performed by a number of different algorithms (Gaussian, Cubic Spline, Fourier transform etc.), often with different

results. Similarly, some operators apply to different geometry types and will need to be implemented by different algorithms. Likewise some algorithms specialise in different feature types such as buildings (Guercke and Sester, 2011). Thus these details need to be formally defined so that automatic selection and execution is possible by the on-demand system.

Formalisation of knowledge can lead to the discovery of new knowledge as long as appropriate formalisation tools are available (Kilpeläinen, 2000). One such tool is the ontology: the explicit specification of the objects, concepts and the relationships in a body of knowledge concerning a particular subject or domain (Gruber, 1993). Ontologies have the advantage of allowing the sharing and reuse of formalised knowledge (Gruber, 1993). Rule-based systems hold *procedural knowledge* that describes explicitly how a process is to be performed. As described earlier (section 1.1), rule-based systems are likely to suffer from rule explosion. Ontologies can hold *declarative knowledge*. One advantage of declarative knowledge is that it can be extended by means of reasoning which can be used to derive additional knowledge (Genesereth and Nilsson, 1998).

The application of ontologies to generalisation is not new. However, to date, their use has been restricted to aiding the process of generalisation, for example by pattern identification (Lüscher et al., 2007); by describing geographical relationships (Dutton and Edwardes, 2006); and by semantically enhancing a line simplification algorithm (Kulik et al., 2005). However, what is proposed is using ontologies to describe the complete process of generalisation. The intention is to formalise the *why*, *when* and *how* of generalisation (McMaster and Shea, 1992).

3.2. Designing the ontology

The first stage in designing an ontology is to determine its scope by defining a set of competency questions that the ontology is expected to answer (Noy and McGuinness 2001). In the domain of on-demand mapping the competency questions include: Under what conditions is generalisation required? Which generalisation operators should be applied? What algorithms should be applied to implement the selected operators? The next step is to enumerate the important terms in the domain.

There are a number of reasons *why* a set of geographic features should be generalised but, if we consider *legibility* in the first instance, we can define a number of *geometric conditions*, such as *congestion* and *imperceptibility*, which are the result of a change from large (detailed) scale to a small scale, and govern legibility. These conditions can be evaluated by applying a number of *measures* (Stigmar and Harrie, 2011). For example, the existence of congestion can be determined by applying a feature *density* measure and will determine *when* generalisation is necessary.

The *how* of generalisation is answered by generalisation operators. But how are they to be defined and classified given the disparate taxonomies described earlier? A (loose) analogy with medicine can be applied. Consider the congestion of features. Congestion can be regarded as a *condition* and a *symptom* of that condition is a high feature density. To check whether the data has that condition a *measure* algorithm can be applied (where a measure algorithm is analogous to a thermometer, say). If the condition is present then a *remedy* such as a reduction in feature size or in feature count is appropriate. Generalisation operators are defined by the remedies they implement. For example, if a set of buildings features is determined to be congested then a reduction in feature count can be applied by the *Selection-ByAttribute* of the more important buildings only or by the *Amalgamation* of buildings into single features. The mechanism by which an algorithm can be used to resolve a condition is shown in Figure 3. That mechanism applied to the particular case of congestion in point data, is shown in Figure 4. In this case the ontology identified two operations, Amalgamation and Selection, as candidates for resolving the congestion. The ontology uses the term *transformation* algorithm instead of generalisation algorithm since the only distinction is made between algorithms that measure and those that transform be it by generalisation or other means.

Operators may also have specific requirements. For example, *Selection-ByAttribute* requires the source dataset to have an attribute that is used to rank the importance of features (such as the severity of road accidents). If this attribute is not present then the operator can be ruled out. All of this

knowledge, in combination, will aid the automatic selection of appropriate operators.

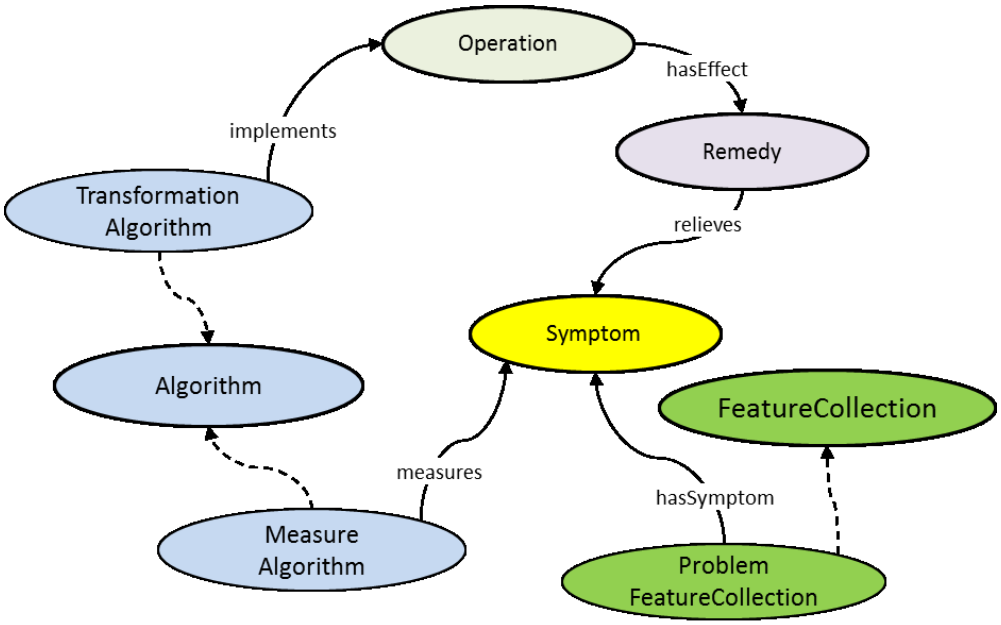


Figure 3 Selecting generalisation algorithms - general case

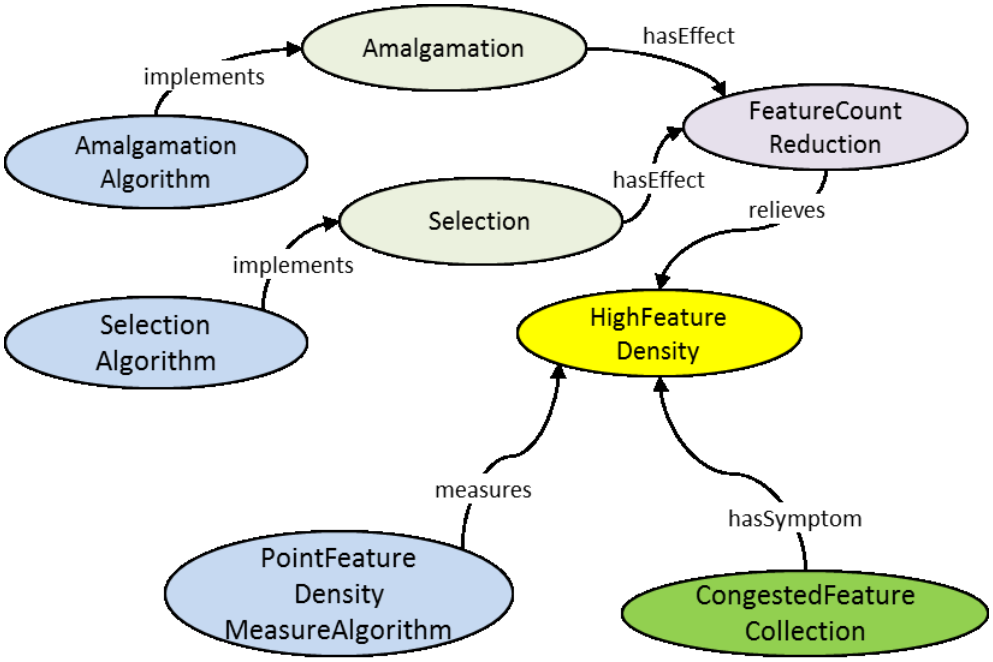


Figure 4 Selecting generalisation algorithms - particular case

However, the domain of the ontology does not stop at the level of the operator. As discussed earlier there is no one-to-one mapping between an operator and the algorithm it implements. Quite different algorithms are required to *Simplify* line features, such as roads, and area features, such as buildings. The ontology is required to have a sufficiently detailed description of generalisation algorithms to allow the relevant algorithm to be selected automatically.

An ontology consists of assertions. We assert that HighFeatureDensity is a Condition of Congestion and that Congestion is a barrier to Legibility. We can also assert that HighFeatureDensity can be remedied by Amalgamation. The advantage of using ontologies is that we can use *reasoning* to *infer* further knowledge. This is the additional knowledge described earlier. For example, we do not have to explicitly state that Amalgamation can resolve congestion. If we sufficiently describe the operators we can infer how they can be utilised.

3.3. Applying the ontology

The ontology can be created in a tool such Protégé which creates and edits OWL (Web Ontology Language) files. Protégé also employs reasoners such as HermiT that can be used to derive inferences. But once developed, how can the ontology be applied in a distributed web service-based system? The application of semantically described geoprocessing services in Spatial Data Infrastructures (Brauner, 2011) provides a parallel.

The standard for implementing geospatial web services is the OGC's Web Processing Service (WPS) protocol and, as described earlier, generalisation algorithms have been implemented with this protocol. The protocol defines a *GetCapabilities* interface that will return a list of each individual spatial operation that the service provides and a free text description of each operation. Also defined in the protocol is a *DescribeProcess* interface that merely describes the input parameters the specified operation requires and its outputs. However, the protocol does not provide for semantic interoperability (Janowicz et al., 2010); that is, there is no method of providing machine readable descriptions of the operations that allow the operation to be selected automatically. What is required is a technique, semantic annotation, that provides these descriptions (Lemmens et al., 2007; Maue et al., 2009; Mladenic et al., 2011).

One solution is the *Semantic Enablement Layer* (Janowicz et al., 2010) where a Web Ontology Service injects semantics into both data and processing service descriptions. A Web Reasoning Service can then be used to match a geoprocessing service to a dataset. Their architecture is aimed at geoprocessing in general rather than generalisation but can be expanded to firstly finding an appropriate measure algorithm for the source data and then, if required, finding an appropriate generalisation algorithm for the existing condition. The selection of the appropriate generalisation operator would be bypassed by inference. That is, if a particular operator remedies a particular condition and a particular algorithm implements that operator then we can infer that the algorithm will remedy the condition.

The ontology can be regarded as component of the semantic referential described earlier (section 1.2) and it extends the Generalisation Knowledge

Ontology of Collagen (section 2.1.3) by expanding the description of the operations. In summary, the use of ontologies to aid the selection of geoprocessing services that implement a specified operation (e.g. buffer) is a well researched area; our contention is that the use of ontologies can be extended to the selection of the (generalisation) operation itself.

4. Foerster – Live Geoinformation with Standardized Geoprocessing Services.

Live geoinformation is considered to be crucial for applications in which decisions a) are based on massive volume of data and b) need to be carried out near real-time (as soon as the data is available). For instance in risk management scenarios, live geoinformation can directly support time critical decision making for saving human lives and infrastructure. Other examples are near real-time analysis of crowd-sourced geodata. All these applications are framed by the idea of the Digital Earth (Gore, 1998) which provides an integrated platform for accessing different kinds of distributed data in near-real time.

Providing such information and transforming raw data into value-added information is supported by geoprocessing. Generalization is involved in any task, in which the scale of the information is affected and is thereby chosen as a representative example. Currently, generalization processes as well as their output (maps, raw data) become available through web service interfaces. These web service interfaces are currently designed along a sequential request-response mechanism, in which the data is sent to the service, processed and then sent back. These different phases are handled sequentially, which means, that the service and the client remain idle in the meantime and wait for the other party to complete. This is not sufficient for live geoinformation and its emerging requirements:

- Performance – Using the idle time of the service, while transferring data.
- Handling, processing, creating of geodata streams – Streams of geodata such as provided by sensors become a valuable source of information for GIS and Digital Earth.

- Loss-less reliable encoding and transfer of data (in contrast to existing lossy unreliable multi media encodings) – The data need to be transferred in a reliable manner, guaranteeing data completeness.
- Interoperability & portability – The data needs to be transferred in an interoperable way by reusing exsting standards and agnostic of the technical setup.

To meet these requirements and realize live geoinformation, HTTP Live Streaming as a loss-less format for real-time data streaming has been combined with the OGC Web Processing Service, which is an established web service interface and de-facto standard for processing geodata on the web. The presented approach is applied to automated generalization of OpenStreetMap data.

Related work

Live geoinformation is about providing information as soon as it is available to the user. This is extremely important for Digital Earth, in which several resources are accessible through a common interoperability layer (Grossner, Goodchild, & Clarke, 2008). To draw appropriate conclusions in time-critical scenarios (e.g. crisis management) from the available data, most up-to-date geoinformation needs to be available. Consequently, one of the backbones of live geoinformation is the extensive use of web technologies to provide the user instantly with information (anywhere, anytime). Therefore, live geoinformation needs to be developed based on current web technologies.

Overall, live geoinformation imposes requirements to data collection, data communication and data integration. These key requirements are high resource utilization rates, simplicity, interoperability and usability. For this article, data integration and data communication are considered from a computational perspective.

Building blocks of live geoinformation are efficiently creating and handling live geodata streams, as applied in this paper. In the context of geoprocessing services, the real-time processing of live geodata streams and publishing such streams is required. Moreover, detecting and extracting events from such geodata streams is highly interesting in the context of Complex Event Processing (Everding, Echterhoff, & Jirka, 2009). Finally, live geoinformation requires a scalable event- and streaming-based architecture for supporting Digital Earth in the future. Regarding the communication within the architecture, we envision a fully push-based architecture, in which

the processes are triggered from the sources (e.g. sensors or created by events). This will limit the communication overhead to a minimum. Technically, this is realized through notification and call-back methods.

Several approaches for improving the scalability and performance of geoprocessing services have been described (e.g. applying Cloud and Grid Computing infrastructures (Baranski, Foerster, Schäffer, & Lange, 2011; Baranski, 2008; Di, Chen, Yang, & Zhao, 2003; Lanig, Schilling, Stollberg, & Zipf, 2008) or the mobile code paradigm (Müller, Bernard, & Brauner, 2010)). Scholten, Klamma, & Kiehle (2006) identify caching, network adaptation, data granularity and communication modes (synchronous vs. asynchronous) as performance criteria.

From a generalization perspective, the work of (Bertolotto & Egenhofer, 2001; van Oosterom, 2005; Buttenfield 2002) about progressive transfer addressed a related problem from a users´ perspective. Users want to receive the data, which is more important first. This is handled by extracting the most important aspects of the data by automated generalization and providing it successivly. Enabling this user experience on the web, a streaming-based processing approach may be suitable.

Approach for Streaming-based Processing

When the WPS receives an asynchronous Execute request, an Execute response is instantly returned to the client and the process execution is scheduled in the background. The Execute response includes a 'Status' element that contains information about the overall status of the process ('accepted', 'started', 'paused', 'succeeded' or 'failed') and an (optional) progress indicator showing the percentage rate of process completion. Furthermore, the Execute response includes a 'statusLocation' element that links another Execute response, which always contains the latest status information about a process. As soon as a process has completed, this Execute response contains the process result(s). The client can constantly pull this Execute response until the final result is available.

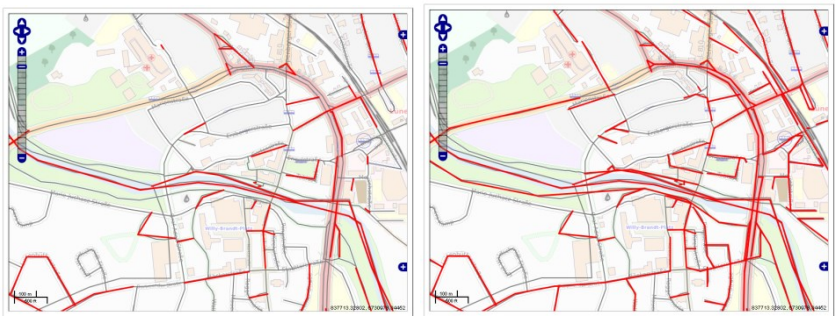
In the proposed approach, the body of the 'Status' element includes an URL to a playlist file as specified by the HTTP Live Streaming draft specification instead of indicating detailed information about the progress of the process as in current WPS implementations (e.g. the amount of features that have been processed). Listing 1 demonstrates an example of an Execute response containing a reference to a playlist file. The format of the playlist file is described further in Section 3.4.

```
<ExecuteResponse service="WPS" version="1.0.0"
statusLocation="...">
 <Process ns:processVersion="1.0.0">
  <Identifier>StreamDouglasPeuckerAlgorithm</Identifier>
 </Process>
 <Status creationTime="...">
  <ProcessStarted>
    http://host:port/wps/playlist?id=123&pollingRate=1
  </ProcessStarted>
```

Listing 1. Exemplary Execute response with an URL of a playlist that contains real-time intermediate results.

The playlist file contains a sorted list of URLs that represents previous and current intermediate results. When an intermediate result is created and stored by the service, the service also updates the playlist file (an URL returning the latest intermediate result is attached). Therefore, by frequently calling the playlist file URL the client receives the latest intermediate results. As soon as a process is completed, the service adds a special tag to the playlist file accordingly. By not adding such a tag, the client knows that the process might run continuously. Further details such as the playlist format encoding, the implementation are described in (Foerster et al 2012).

Streaming-based Processing for Automated Generalization



t1 t3

t2 t4

Discussion & Related Challenges for Generalization

As described, Web Services face a challenge of providing the most-current data as soon as it is available. Mostly, this data needs to be adopted regarding scale through processing, so generalization is a challenge. To tackle this challenge, a loss-less, asynchronous stream-based and interoperable approach towards web service interfaces is required. The presented approach is based on HTTP Live Streaming and is applied to generalization of Open Street Map data. From the research it becomes evident, that not all generalization processes are suitable for this approach. If the context of the objects and their topology play a significant role in the specific process, it cannot be applied to streaming-based processing. The size of the data partition must not exceed the size of the transferred data chunks. However, tasks of simplification as for instance Douglas Peucker, which are still heavily used in generalization batch jobs appear to be suitable candidates.

Future research needs to address these issues in more detail and find intelligent ways to enable streaming-based processing for a broader range of generalization functionality. This can be done by automatically detecting the partition requirement and adjust the size of transferred data chunks accordingly. As soon as this is achieved, the chaining and orchestration of these streaming-enabled generalization processes becomes an interesting application, to deliver highly customized datasets in a timely fashion.

5. Conclusion

We have seen in this chapter that modelling the process of generalisation is still a very open question, and continues to generate interest in the research

community. While a lot of generalisation tools have been studied and implemented, the challenge of creating systems capable of using them automatically is still alive. We have seen that different models have been studied and automated solutions have been successfully implemented, but these successes have been limited to building systems that provide a static solution to derive a specific product from a specific set of data.

In order to overcome this limitation, it has long been recognised that studies needed to concentrate on the process modelling side. But in order to do this successfully, and to be able to test the concepts, we had to have the basic tools (generalisation operators, measures) readily available. Web services have been proposed as a way to encapsulate and publish these tools, so that they can be easily reused.

Several studies now focus on "on demand mapping", which looks into designing systems capable of interpreting user requirements (in the form of formal machine readable specifications, see chap.2), and deriving the appropriate map. This requires a significant effort in formalising the description of all the components of the system: specifications, data, tools, knowledge. The Collaborative Generalisation approach described in the case study 1 of this chapter, shows how to different apply existing models in different situations occurring on the same map. The second case study presented an ontology based approach to resolve cartographic conflicts as they occur during the automatic creation of a map. Both these approach rely on formalising aspects of the generalisation knowledge. Both also rely on existing software components providing the lower functionalities of the system (operations, measures, or a particular subsystem to perform a specific generalisation task). As it is often suggested that such decoupled system should use Web Services to access the lower functionalities, we have include a third case study focusing on improving the performance of web processing services, using the concept of streaming based processing.

The next challenges in the area of process modelling related to generalisation are related to the design of systems able to perform on demand mapping, which includes generalisation, but also the collection of user requirements, data integration and automatic cartographic design (to style the resulting map). As prototypes of these systems emerge, we know that they will be hindered by performance issues. This is due to the fact that the lack of predefined sequence of action will be overcome by complex strategies to build the sequence, possibly including expensive trial and error strategies. More research in optimisation techniques will therefore be required (machine learning, parallel computing, streaming based processing, etc.)

References

Alameh N (2003) Chaining Geographic Information Web Services. IEEE Internet Computing 07:22−29

Armstrong MP (1991) Knowledge Classification and Organization. In: Buttenfield B, McMaster RB (eds) Map Generalization: Making Rules for Knowledge Representation. Longman, London, p86-102

Balley S, Jaara K, Regnauld N (2012) Towards a Prototype for Deriving Custom Maps from Multisource Data. 15th Workshop of the ICA Commission on Generalisation and Multiple Representation, Istanbul, September 2012

Balley S, Regnauld N (2011) Models and standards for on-demand mapping. In Proceedings pf the 25th International Cartographic Conference. Paris, July 2011.

Brauner J(2012) Ad-hoc-Geoprocessing in Spatial Data Infrastructures – Formalizations for Geooperators. In Bernard L and Pundt H (eds), 1st AGILE PhD School. Shaker Verlag, Wernigerode, Germany.March 2012.

Burghardt D, Neun M, Weibel R (2005) Generalisation Services on the Web – A Classification and an Initial Prototype Implementation. Cartography and Geographic Information Science, 32(4):257-268

Burghardt D, Petzold I, Bobzien M (2010) Relation modelling within multiple representation databases and generalisation services. The Cartographic Journal 47 (3):238-249

Bader M, Barrault M, Weibel R (2005) Building displacement over a ductile truss. International Journal of Geographical Information Science 19(8):915-936

Brenner C, Sester M (2005) Cartographic generalization using primitives and constraints. In Proceedings of the 22nd International Cartographic Conference, A Coruna, Spain, July 2005

Dean J, Ghemawat S (2004) MapReduce: Simplified Data Processing on Large Clusters. In Proceedings of the Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December 2004

Douglas D, Peucker T (1973) Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. The Canadian Cartographer 10(2):112-122.

Dutton G, Edwardes A (2006) Ontological Modeling of Geographical Relationships for Map Generalization. In Proceedings of the 9th Workshop of the ICA Commission on Map Generalization and Multiple Representation, Portland, USA

Edwardes A, Burghardt D, Bobzien M, Harrie L, Reichenbacher T, Sester M, Weibel R (2003) Map Generalisation Technology: Addressing the Need for a Common Research Platform, In Proceedings of th 21st International Cartographic Conference, Durban, South Africa, p. 170-179.

Foerster T, Stoter J, Köbben, B (2007) Towards a formal classification of generalization operators. In Proceedings of the 23rd International Cartographic Conference, Moscow, August 2007

Foerster T, Stoter J, van Oosterom P (2012) On-demand base maps on the web generalized according to user profiles. International Journal of Geographical Information Science 26(1): 99-121

Genesererh MR, Nilsson NJ (1998) Logical foundations of artificial intelligence. Morgan Kaufmann Publishers. Palo Alto, California

Gruber T (1993) A translation approach to portable ontology specifications. Knowledge Acquisition 5(2):199-220

Guercke R, Sester M (2011) Building Footprint Simplification Based on Hough Transform and Least Squares Adjustment. In Proceedings of the 14th Workshop of the ICA commission on Generalisation and Multiple Representation. Paris, France

Harrie LE, Sarjakoski T (2002) Simultaneous graphic generalization of vector data sets. Geoinformatica 6(3):233-261. doi:10.1023/A:1019765902987

Jäger E (1991) Investigations on Automated Feature Displacement for Small Scale Maps in Raster Format. In Proceedings of the 15th International Cartographic Conference, p 245-256

Janowicz K, Schade S, Broring A, Kessler C, Maue P, Stasch C (2010) Semantic Enablement for Spatial Data Infrastructures. Transactions in GIS 14(2):111

Kilpelainen T (2000) Knowledge Acquisition for Generalization Rules. Cartography and Geographic Information Science 27(1):41-50

Kulik L, Duckham M, Egenhofer M (2005) Ontology-driven map generalization. Journal of Visual Languages & Computing 16(3):245-267

Jenks GF (1989) Geographic logic in line generalisation. Cartographica 26 (1):27-42.

Lemmens R, de By RA, Gould M, Wytzisk A, Granell C, van Oosterom P (2007) Enhancing Geo-Service Chaining through Deep Service Descriptions. Transactions in GIS 11(6):849-871

Lüscher P, Burghardt D, Weibel R (2007) Ontology-driven Enrichment of Spatial Databases. In Proceedings of the 10th Workshop of the ICA commission on Generalisation and Multiple Representation. Moscow, Russia, August 2007

Lutz M (2007) Ontology-Based descriptions for semantic discovery and composition of geoprocessing services. GeoInformatica 11(1):1-36. doi:10.1007/s10707-006-7635-9

Mackaness WA, Edwards G (2002) The importance of modelling pattern and structure in automated map generalisation. In: Proceedings of the Joint ISPRS/ICA Workshop on Multi-Scale Representations of Spatial Data, p 7-8

Maue P, Schade S, Duchesne P (2009) Semantic annotations in OGC standards Open Geospatial Consortium.

McMaster RB, Shea KS (1992) Generalization in digital cartography. Washington DC, Association of American Geographers.

Mladenic D, Moraru A, Skrjanc M (2011) Deliverable D4.2 Model Annotation component and guidelines. Envision Consortium. http://www.envision-project.eu/2011/01/deliverable-4-2-model-annotation-components-and-guidelines. Accessed 15 March 2013

Müller M, Bernard L, Brauner J (2010) Moving Code in Spatial Data Infrastructures – Web Service Based Deployment of Geoprocessing Algorithms. Transactions in GIS 14(S1):101–118

Müller M, Kadner D, Bernard L (2012) Moving Code – Sharing Geospatial Computation Logic on the Web. In Proceedings of the 15th AGILE International Conference on Geographic Information Sciences, Avignon

Neun M, Burghardt D (2005) Web Services for an Open Generalisation Research Platform. In Proceedings of the 8th ICA Workshop on Generalisation and Multiple Representation, A Coruna, Spain

Noy NF, McGuinness D (2001) Ontology Development 101: A Guide to Creating Your First Ontology. Stanford University. http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf. Accessed 15 March 2013

Regnauld N, McMaster RB (2007) A synoptic view of generalisation operators. In Mackaness WA, Ruas A. and Sarjakoski LT (eds.) Generalisation of Geographic Information. Elsevier Science B.V, Amsterdam, p 37-66

Rieger MK, Coulson MRC (1993) Consensus or confusion: cartographers' knowledge of generalization. Cartographica: The International Journal for Geographic Information and Geovisualization 30(2):69-80

Ruas A, Duchêne D (2007) A prototype generalisation system based on the Multi-Agent system paradigm. In: Mackaness WA, Ruas A, Sarjakoski LT (eds) The Generalisation of Geographic Information : Models and Applications, Chapter 14. Elsevier, p 269-284

Ruas A, Plazanet C (1996) Strategies for automated generalization. In: 7th International Symposium on Spatial Data Handling, Delft, Netherlands, p 319-336

Schaeffer B, Foerster T (2008). A Client for Distributed Geo-Processing and Workflow Design. Journal for Location Based Services 2(3):194-210.

Stigmar H, Harrie L (2011) Evaluation of Analytical Measures of Map Legibility. The Cartographic Journal 48(1): 41-53

Stoter J, Burghardt D, Duchêne C, Baella B, Bakker N, Blok C, Pla M, Regnauld N, Touya G, Schmid S (2009) Methodology for evaluating automated map generalization in commercial software. Computers, Environment and Urban Systems 3(5):311-324

Taillandier P, Duchêne C, Drogoul A (2008) Knowledge revision in systems based on an informed tree search strategy: application to cartographic generalisation. In Prodeedings of

the International Conference on Soft Computing as Transdisciplinary Science and Technology (CSTST), Cergy-Pontoise, France, July 2008

Taillandier P, Duchêne C, Drogoul A (2011) Automatic revision of rules used to guide the generalisation process in systems based on a trial and error strategy. International Journal of Geographical Information Science 25(12):1971-1999

Taillandier P, Gaffuri J (2012) Improving map generalisation with new pruning heuristics. International Journal of Geographical Information Science 26(7):1309-1323

Touya G (2008) First thoughts for the orchestration of generalisation methods on heterogeneous landscapes. In: Proceedings of 11th ICA Workshop on generalisation and multiple representation, Montpellier, France

Touya G (2010) Relevant space partitioning for collaborative generalisation. In: Proceedings of 12th ICA Workshop on Generalisation and Multiple Representation, Zurich, Switzerland

Touya G (2012) Social welfare to assess the global legibility of a generalized map. In: Xiao N, Kwan MP, Goodchild MF, Shekhar S (eds) Geographic Information Science. 7th International Conference, GIScience 2012, Columbus, USA. Lecture Notes in Computer Science, vol 7478. Springer, Heidelberg, p 198-211

Touya G, Duchêne C (2011) CollaGen: Collaboration between automatic cartographic generalisation processes. In: Ruas A (ed) Advances in Cartography and GIScience. Lecture Notes in Geoinformation and Cartography, vol 1. Springer, Heidelberg, p 541-558

Touya G, Duchêne C, Ruas A (2010) Collaborative generalisation: Formalisation of generalisation knowledge to orchestrate different cartographic generalisation processes. In: Fabrikant S, Reichenbacher T, van Kreveld M, Schlieder C (eds) Geographic Information Science. Lecture Notes in Computer Science, vol 6292. Springer, Heidelberg, p 264-278