1Generalise: 1Spatial's new automatic generalisation platform

Nicolas Regnauld
Nicolas.regnauld@1spatial.com
Product Manager
1Spatial
Tennyson House, Cambridge Business Park
Cambridge, CB4 0WZ
United Kingdom

Introduction

In the past few years, a number of map production systems using a high degree of automated generalisation have been built. This was made possible by all the efforts made during the past two decades in researching the domain of automated generalisation, and in taking some of these research ideas into commercial or open source systems. Chapter 11 of (Burghart et al 2014) describes the main successes achieved in the domain of automated generalisation in various national mapping agencies. See also (Regnauld et al 2013),(Stoter et al 2013) and (Maugeais 2011) for more information on systems in place at Ordnance Survey GB, The Dutch Kadaster and IGN France. These systems all process high resolution topographic data and create derived data at around 1:25-50k scales, sometimes also producing Digital Landscape Models along the way.

While these systems represent a landmark in the quest for automated generalisation, they do not signal the end! There is one big commonality between these systems: they required very heavy investments to build them, as they are all bespoke systems, even if they have been built on top of commercial software. As the software provider of the platform behind a few of these systems (IGN France, OSGB, ADV German Landers), 1Spatial has realised that the next challenge for the software providers is to propose a platform that is quick and easy to use, and can still be configured to deliver the data products needed. 1Spatial has therefore decided to invest in the development of 1Generalise, a new platform dedicated to automated generalisation.

1Generalise, a component of the 1Spatial Management Suite (1SMS)

1Generalise is a component of 1SMS (Figure 1). This suite of software covers all the steps required by National Mapping and Cadastral Agencies (NMCAs) to plan jobs for collecting data, maintain a spatial database, and then derive products from that database. 1Generalise sits close to the end of the chain, and focuses on exploiting data contained in a spatial database to derive less detailed products, typically to derive maps at a smaller scale.

1Generalise is a server-based product accessed either through a browser-based user interface or through an automatic workflow triggering generalisation jobs through a SOAP API. The server contains a controller which can use a grid of processing nodes to execute generalisation jobs in parallel. This brings scalability to the system, which can be expanded simply by adding nodes to the grid to process large numbers of jobs more efficiently.

The current prototype of 1Generalise supports reading spatial data from an Oracle database, and this will be extended to other common formats for version 1.0

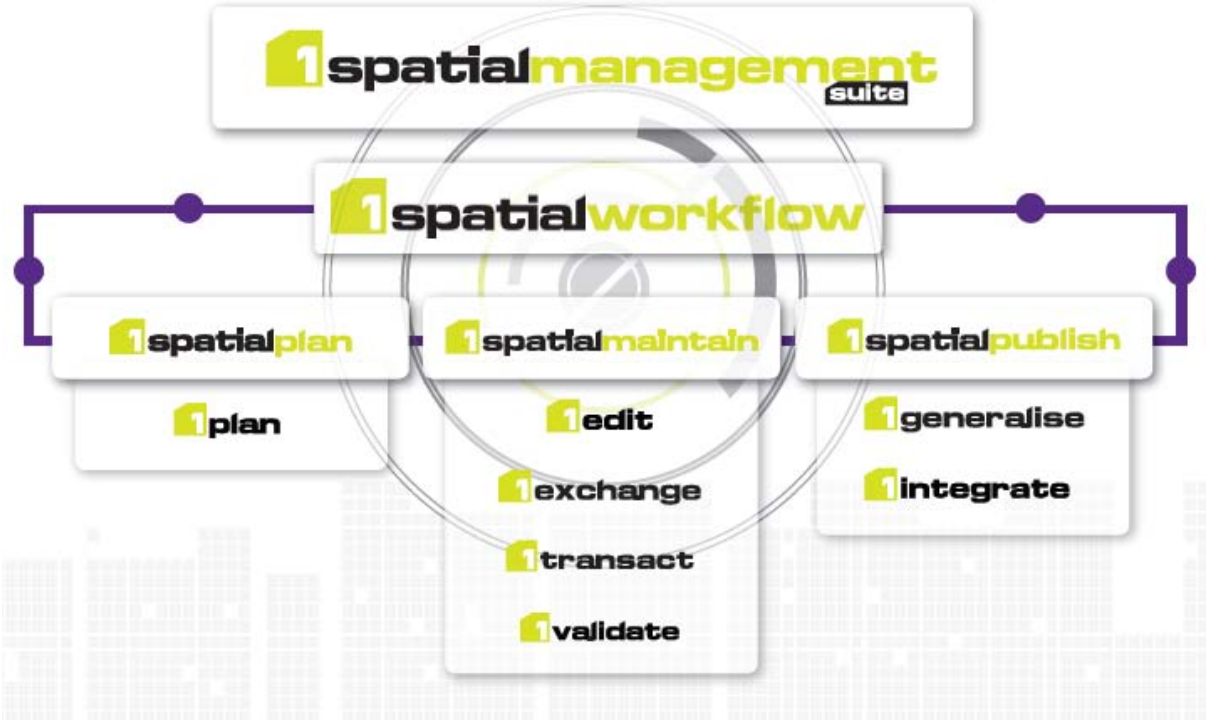

Figure 1: 1Generalise, a component of the 1Spatial Management Suite

Reusing the rich generalisation heritage

Over the last 20 years, 1Spatial has developed exciting technologies for the automation of generalisation. Gothic, the data cache that is used in most components of 1SMS, is a versioned Object Database, that handles geometric features, explicit topology, spatial indexing, spatial queries and a provides a variety of geometric tools to measure and transform geographic features. It also benefits from a multi agent framework with an optimisation engine to facilitate the expression and resolution of complex contextual generalisation problems, using constraints. This multi agent framework evolved from the results of the European project AGENT (Barrault et al 2001). 1Generalise includes a web services API that provides interfaces for integration into larger systems, and a flexible processing environment which can easily be scaled up by adding more processing nodes to the grid. Figure **2** shows the high level architecture of the system.

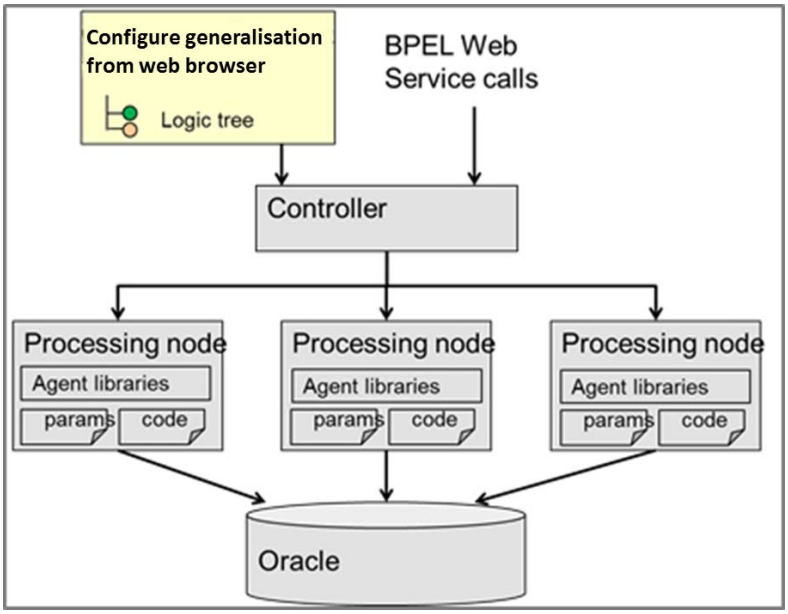

Figure 2: High Level Software Architecture

Main concepts

1Generalise does not look like a traditional GIS where the main screen provides a viewer for the data from which tools can be accessed through toolbars. 1Generalise is an application designed to support setup and running of generalisation processes on a set of data, before exporting the results to an output dataset. This new dataset can then by styled using map production tools or data servers in order to produce output maps.

In order to understand how 1Generalise works, the main concepts of the system need to be understood:

- **User classes**: 1Generalise requires that all the features that need to be recognisable in the output dataset must be in separate input classes (if hospital buildings and school buildings need to be styled differently on the target map, 1Generalise needs them in separate input classes). 1Generalise provides an interface to help the user define these input classes, and define the rules to import the data into them. This needs to be done for the input classes, and for the output classes. There is no restriction on the schema used to store the user data outside the system.
- **System schema**: 1Generalise has an internal schema with default classes that represent typical geographic feature types (Roads, buildings, landcover, etc.), or other spatial concepts that may be required during the generalisation process (road partition, urban area, coastal area, etc.). These classes which we'll call system classes are associated with Subflows, specifying how they will be generalised.
- A **Subflow** is attached to a class of the internal schema, it contains a sequence of actions that is applied to each of the instances of the class. A class can have several Subflows, used in different Flowlines or in different places in the same Flowline. A Subflows exposes all the parameters required by the actions it contains.
- A **Flowline** is an ordered list of Subflows, it defines a full generalisation process. A Flowline can be parameterised.

- The **Parameters** available for tuning a Flowline is the aggregation of all the parameters required by the actions used in the Subflows involved in the Flowline. There are two types:
    - Global parameters, which have a single value for all classes
    - Class based parameters which can have different values for different classes. Each Flowline provides default values for all parameters
- The P**rofile** is where the user chooses the generalisation Flowline, source and target datasets to use, and where they associate their input classes with the system classes and the output classes. Once this is done, the system is able to identify all the class parameters required, give them default values and let the user modify them.
- The **Job** is the last component. Once a Profile is set up, a Job can be created to run the profile on a chosen extent within the input dataset. A Job can be triggered manually through the Web browser client, or a SOAP API. When executing a Job, 1Generalise clears the extent of the Job in the target dataset, loads the source data for that extent, generalises it and exports the result to the target dataset.

1Generalise Interfaces

1Generalise has three interfaces:

- The **User Interface**, accessible through a web browser, allows the user to apply an existing Flowline to their data and get the result in a new dataset. Figure 3 shows the current version of the main screen for the user interface. It contains the following sections:
    - The *Connections* section allows the user to specify where the data is located and how to access it. Several connections may be necessary, as the data can be read from multiple databases and the output exported to a different one.
    - The *Classification* section allows the user to define how their data will be imported into the system (reclassification rules based on attributes). They need to be imported into classes that can later be mapped to classes of the internal schema. In a similar way, it also lets the user define how the result will be exported to the target dataset. Each classification is based on a single connection.
    - *Flowlines* let the user see the content of all the Flowline available, so that they can choose the one they want to use.
    - The *Profiles* section let the user create and modify their Profiles. In a Profile, the user chooses where the data come from through the choice of input and output classifications. The user also chooses which Flowline they want to use (see Figure 4). Once this is set, an interface lets the user setup the mapping between each input class, the corresponding output class and the system class that will provide the generalisation capabilities. This is illustrated in Figure 5, where a single mapping has been set up to generalise features of the “Input Building” class into features of the “Output Building” class, using the Subflow “060 Buildings”, associated with the internal class “Building”. The third and fourth tabs let the user tune the global and class parameters. Global parameters have a single value for a Flowline (like a target scale, a geometric precision, etc.), while class parameters are specific to a class. In the example shown in Figure 6, the five parameters for the OutputBuilding class came from the Subflow “060 Buildings”, if another class of buildings (for example hospital) had been using the same subflow, it would also have the same parameters,

but the user would be able to set different values. All the parameters come with default values that can be changed. Once a Profile has been setup, it can be used to create multiple Jobs.

- The *Job* section allows the user, or an external workflow system, to use a Profile to process a specific extent of the dataset. Once a Job is set up (Profile and extent specified), the user can run the Job, and see all the actions being processed. Each action reports on any failures encountered. Once a Job is complete, a viewer can be opened, showing two windows side by side where the data can be displayed at different stages in the process for comparison. This is useful in understanding what happens at each step of the process, and helps when tuning parameters by allowing the user to check their effect.

Figure 3: 1Generalise User Interface (main screen)

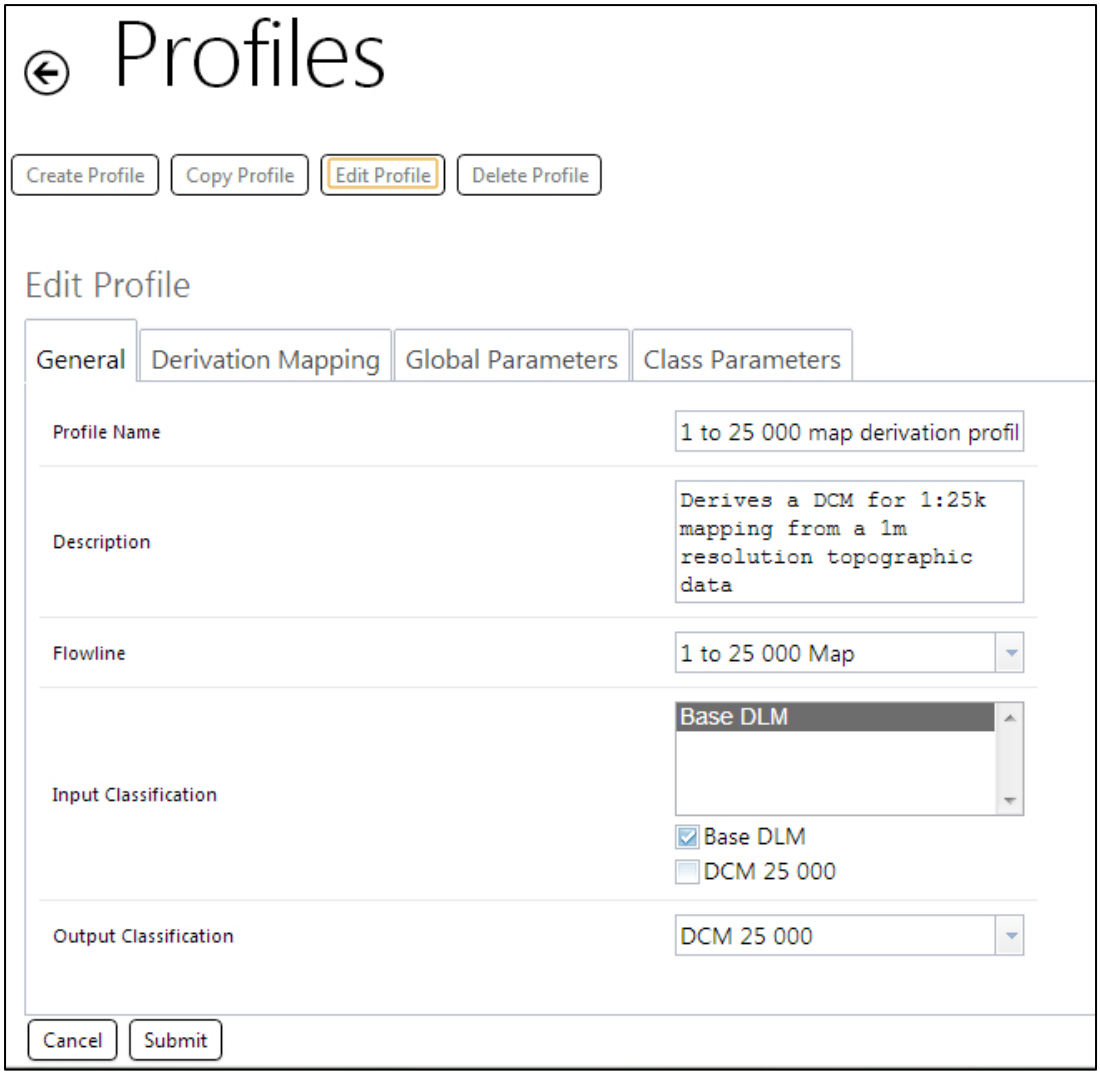

Figure 4: Profile Interface, general tab

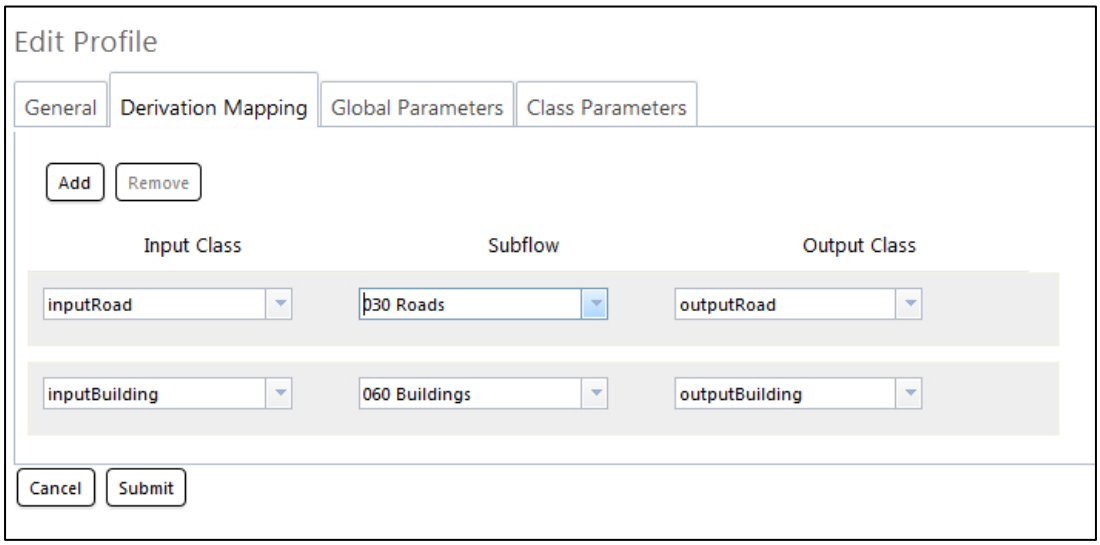

Figure 5: Profile Interface, Derivation Mapping tab

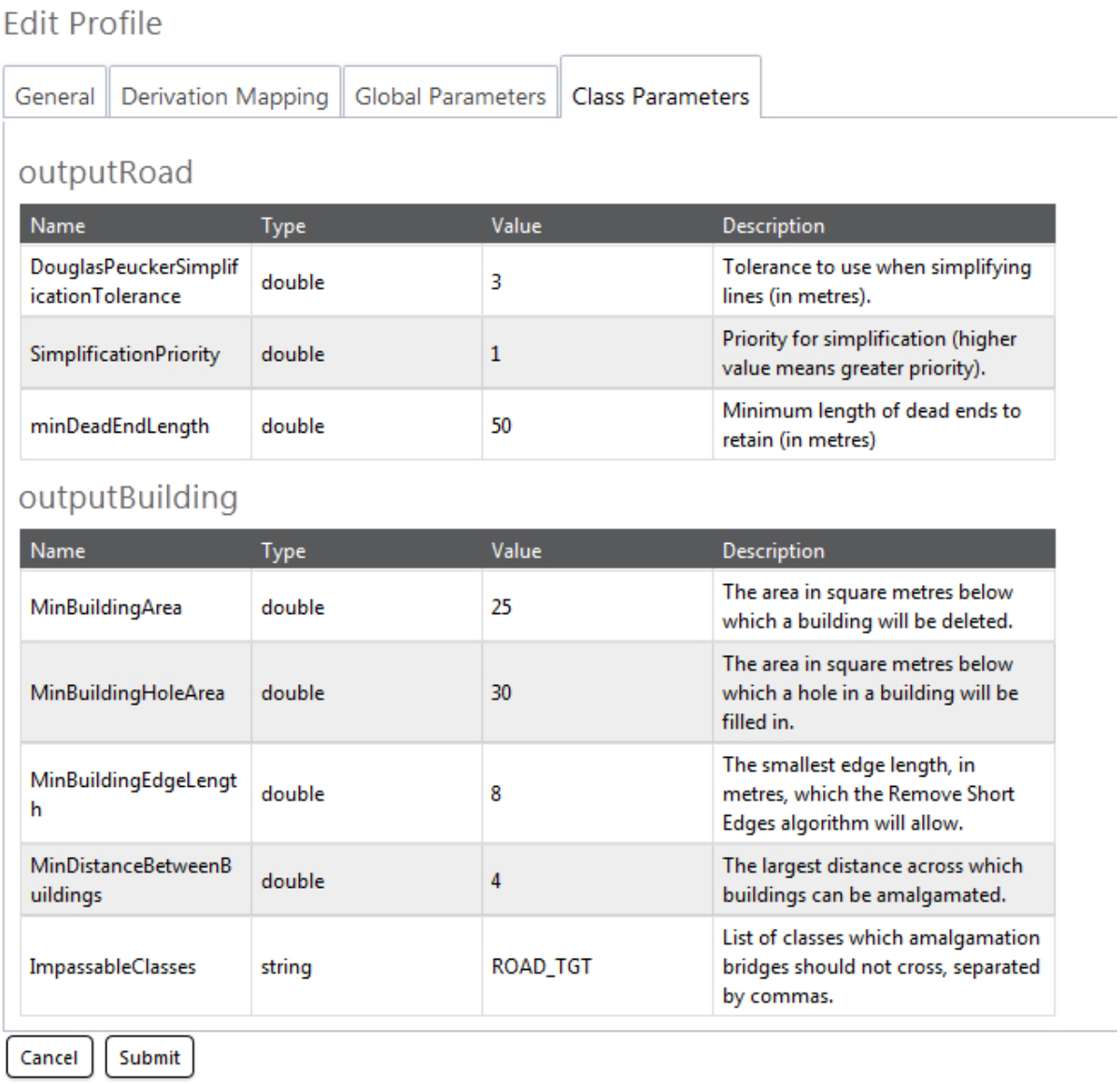

Figure 6: Profile interface, class parameters tab

- The **Flowline designer Interface**, also accessible through a web browser, allows a specialist user to modify Flowlines or create new ones. A Flowline is a sequence of Subflows, where each Subflow is a sequence of actions applied on a class. A Flowline can contain several Subflows referring to the same class, i.e. for example a Subflow to thin the road network could be used early in the process by the Flowline, and another one to resolve proximity conflicts between roads could be used later. The Flowline designer is also able to write rules-based actions, and decide which parameters should be exposed to the User Interface, and what default values to give them.
- The **1Generalise API**, which allows an external system (like a workflow management software), to trigger 1Generalisation jobs.

1Generalise operating modes

1Generalise can be operated in 3 modes:

- **Simple process**

  This is the simplest processing mode. The user creates a Job that contains a Profile and a data extent. When the Job is executed, the results can be viewed in the internal viewer which allows the user to see side by side two views on the data at any two intermediate states in the process. For exemple Figure 7 shows on the left buildings after small ones have been deleted, and on the right buildings after they have been aggregated, but before simplification has occurred (see Figure 8 on the right to see the result after simplification). This is very useful to tune a Flowline.

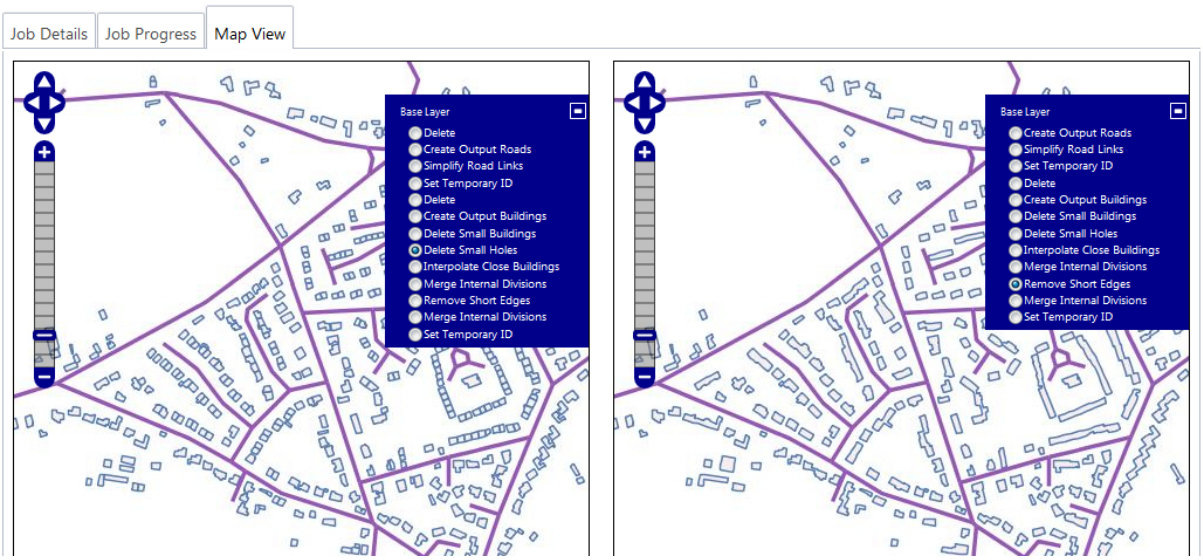

Figure 7: 1Generalise internal viewer

- **Initial Load**

  Based on the size of the dataset and the speed required the simple mode mentioned above may not be adequate. For large Jobs 1Generalise provides a Flowline to partition the initial dataset and create Jobs to generalise each partition using a second Flowline. These jobs can get triggered in parallel using all the nodes available to the controller on the grid.

- **Change only Update**

  1Generalise is also designed to function in a Change only Update (CoU) mode. The principle is that an extent containing some recorded change can be passed to 1Generalise with the Profile required to generalise the data. 1Generalise will automatically create a dynamic partition around that extent to include the context it needs, and define a crisp partition within which the data can be generalised and replaced in the target dataset. For this the Flowline must be built following a specific model.

  - ✓ Step1: identify the partitioning features (set of configurable classes) that need to be processed. If some partitioning features are among the changed features that need to be re-generalised then the system checks that there is enough contextual information to generalise them. For example, if some dual carriageways need

collapsing, the roads connected to them also need to be available. If not, the step makes the system automatically restart the process using a larger extent to pick up more contextual data. This may be done several times until enough data is loaded to find the partitioning features required.

- ✓ Step 2: The partitioning features are cleared from the target dataset, generalised (using a specific Subflow) and stored there.
- ✓ Step 3: Create partitions using the partitioning features.
  The system checks that all the changed data is included in one of the partitions and if not it causes the system to restart using a larger extent, this can happen several times until enough partitioning features are loaded up to form partitions around all the changed data.
- ✓ Step 4: Generalise inside each of the partitions which contain some changed data.
  The first step is to use the generalised boundaries to query the data inside the partition in the target dataset, and clear its content. Then the features inside the partition are generalised and the result committed to the target database.

Figure 8 shows on the left the extent of the update job that was sent to 1Generalise (red square). The system has loaded more data than the extent requested, until it had enough roads to form partitions fully containing the extent of the change. Note that some of the partitions do not intersect with the change extent (displayed in a lighter colour). These are not reprocessed. Only those containing some change data are reprocessed. On the left you can see these partitions at a point in the process where the data in the target dataset had been cleared and the source data loaded into it. The figure on the right shows the result, after the buildings in these partitions have been generalised.

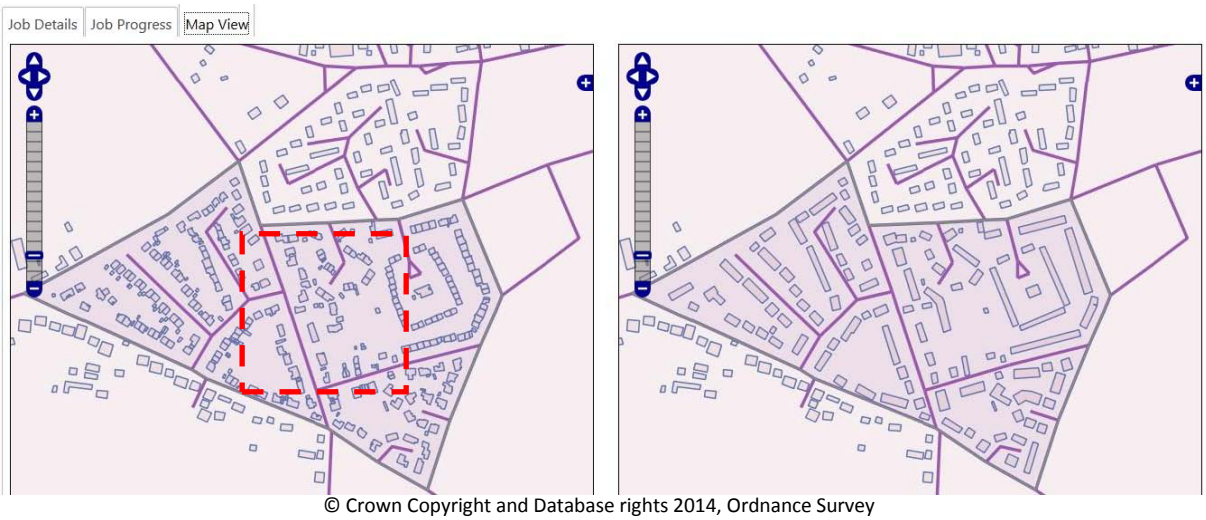

Figure 8: Automatic expansion of the job extent

Potential conflicts between different update jobs can occur. This is due to the expansion of the input extent that 1Generalise applies to have enough context to perform the generalisation. As this is not expected to happen frequently, when a conflict is detected, the second job will be aborted and reprocessed later.

Development of 1Generalise

1Generalise is currently under development using AGILE principles (see the AGILE manifesto (Beck et al 2001)). The developments are conducted in sprints of two weeks, at the end of which a new release of the working software is made. This allows 1Spatial and 1Spatial's customers to test the product from an early stage and provide feedback during development. At the beginning of each sprint a planning session defines the items that will be worked upon during the sprint. These are defined by the product owner and the customer, to prioritise items that are of most value. Value sometimes means focusing on getting early results to prove capabilities of the software, sometimes means de-risking an unproven aspect of the design of the software. Each sprint is followed by a retrospective for the team to learn lessons from the past sprint, and a review where the team demoes the new functionality of the software to the stakeholders.

This approach has worked very effectively, and the development of 1Generalise is making steady progress. The interfaces that we have presented in this paper are purely functional at this stage, allowing the team to test the capabilities of the software. They will be continuously reviewed and improve throughout the development process to improve its usability, to make them as easy and intuitive to use as possible.

Conclusion and future work

1Generalise version 1 is in an advanced stage of development and can already perform complex generalisation tasks. A lot of the developments so far have focused on building the infrastructures of the system to support parallel processing during initial load, incremental updating, automatic reports, easy integration with other systems, and above all a system that is fully and quickly useable out of the box yet highly configurable.

Once version 1 has been released (currently expected to be released towards the end of 2014), the system will be enriched to include more generalisation operators, which will be available for building a wider range of Flowlines. 1Spatial will also include a new product into 1SMScalled 1Publish, which will provide cartographic styling capabilities. 1Publish will come from rebranding the Mercator platform (formerly developed by Mercator CPS, now part of 1Spatial) and developing it to facilitate its integration with other components within 1SMS (1Generalise in particular).This development will allow the use of 1Generalise as an integrated tool for bulk or incremental cartographic generalisation as part of a full data production and product generation system.

References

1Spatial (2014) http://1spatial.com/products-services/1spatial-management-suite

Barrault M, Regnauld N, Duchêne C, Haire K, Baeijs C, Demazeau Y, Hardy P, Mackaness W, Ruas A, Weibel R (2001) Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization. In: Chinese Society of Geodesy Photogrammetry and Cartography (eds) Proceedings of the 20th international cartographic conference, Beijing, 2001

Beck, Kent, et al. (2001). "Manifesto for Agile Software Development". Agile Alliance

Burghardt, Dirk, Duchene, Cécile, Mackaness, William (Eds.) (2014) Abstracting Geographic Information in a Data Rich World, Series: Publications of the International Cartographic Association (ICA).

Maugeais E, Lecordix F, Halbecq X, Braun A (2011) Dérivation cartographique multi échelles de la BDTopo de l'IGN France : mise en œuvre du processus de production de la Nouvelle Carte de Base. In: Proceedings of the 25th International Cartographic Conference, Paris, July 3-8.

Regnauld N, Lessware S, Wesson C, Martin P (2013) Deriving products from a Multi Resolution database using automated generalisation at ordnance survey. In: Proceedings of the 26th International Cartographic Conference, Dresden, 25–30 Aug 2013.

Stoter JE, Post M, van Altena V, Nijhuis R, Bruns B (2013) Fully automated generalisation of a 1:50k map from 1:10k data. Cartography and Geographic Information Science, vol 14, Issue 1.