Workshop of the ICA Commission on Map Generalisation and Multiple Representation – June 25th 2006

Workflow Management and Generalisation Services

Ingo Petzold[1], Dirk Burghardt[1], Matthias Bobzien[2]

[1]GIS Division, Department of Geography, University of Zurich, Winterthurerstrasse 190, 8057 Zurich (Switzerland), Fax: +41-1-635 6848, Email: {petzold,burg}@geo.unizh.ch
[2]Axes Systems AG, Brünigstrasse 12, 6055 Alpnach (Switzerland), Email: m.bobzien@axes-systems.com

KEYWORDS: Workflow, Web Services, Generalisation, Automation

1. Motivation

In the domain of map production two trends are currently noticeable. On the one hand, a continued automation of the processes for derivation of traditional, analogue paper maps, and on the other, functions for delivering dynamic maps in web mapping and mobile applications e.g. on PDAs. In these scenarios identical geo data of one reference scale provides the basis for the specific maps. The derivation is based on generalisation employing different functions. To provide this functionality independent from operating system and programming languages the concept of web services are applied to the field of automated generalisation (Neun et al., 2006; Burghardt et al., 2005a). These allow the integration of external generalisation functionality in existing GIS and map production systems, without changing the developed production runs. Ideas about web services are discussed on the international generalisation workshops of the "ICA Commission on Generalisation and Multiple Representation".

An important scientific question under the topic of cartographic generalisation is the automated selection, chaining and parameterisation of generalisation functions. Until now there are only a few scientific publications treating this problem. One reason is that experiments can only be carried out with an extensive set of generalisation functions, often not available on one platform. Worth mentioning is the AGENT approach (Ruas, 1999; Barrault et al., 2001), whereby the orchestration is carried out with help of meso agents on the basis of cartographic constraints. Further approaches make use of several optimisation methods (least square adjustment, energy minimisation, simulated annealing) to combine different generalisation functions. The development of generalisation services solves the problem of the availability of an extensive set of generalisation functions. The research presented in this paper will focus therefore on the usage of workflow management systems for orchestration and combination of different generalisation services. This proposed approach is currently being implemented in axpand, the cartographic GIS of Axes Sytems (Axes Systems, 2006).

2. Related work

2.1 Examples of existing workflow management systems in GIS

Currently, several workflow management systems exist as part of commercially available GIS. Following two examples are shown in more detail: first the ModelBuilder from ESRI and second the Workbench of FME (Feature Manipulation Engine) from Safe Software. The ModelBuilder from ArcGIS from ESRI (ESRI, 2006) is an example for the realisation of static workflows – sequential flows without choices and loops (see section 4.2). ESRI calls workflows models. The creation of a dynamic workflow (implying choices and loops) is only possible after leaving the GUI of the ModelBuilder application of ArcGIS – export of the workflow and using a script language like Python. For the static workflows variables (parameters) and tasks are visualised by the GUI of the ModelBuilder. Variables are represented by ellipses and are more or less files in which data is stored or loaded from. Tasks represent a tool, an ArcGIS function, which can be selected from the Toolbox. Evaluation functions do not exist in the Toolbox. The input and output parameters of the tasks can be graphically assigned from the variables as well as marked as input or output parameters of the model. An example of a workflow designed with the ModelBuilder is shown in Figure 1.

Another example for utilisation of a workflow management system for the processing of geo data is given by the Workbench of FME from Safe Software (Safe Software, 2006) (Figure 1). The FME Workbench offers a graphical interface to define a workflow for the transformation and manipulation of geo data.

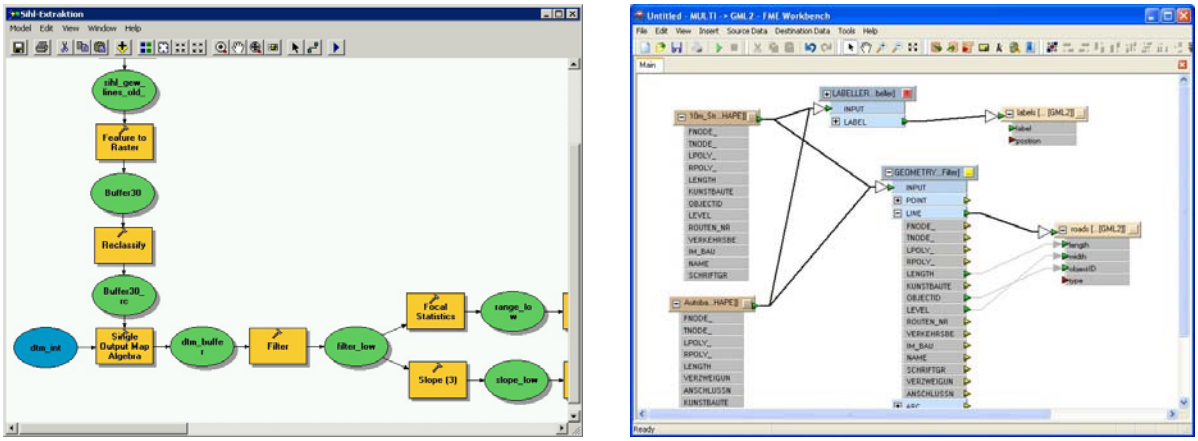

Figure 1. Screenshot of a workflow designed with the ModelBuilder of ArcGIS 9.1 from ESRI (2006) left and with the Workbench of FME 2006 from Safe Software (2006) right.

2.2 Approaches to apply workflows for the generalisation process

The modelling of distributed GIS data production with help of workflows is proposed in Li and Coleman (2005). They showed that workflow technology provides real-time access to status of project progress and support the control of production processes as well as the automation of production procedures. The application of workflows in the generalisation domain is known mainly from sequential batch-processing approaches. An example is shown in Figure 2 (Lee 2003). A more general workflow for generalisation purpose is discussed in Hardy and Lee (2005). There, a distinction is made between a pre-processing steps and a main generalisation loop. The pre-processing step contains data structure enrichment, partitioning, sub-division and classification as well as pattern and group detection. The main generalisation loop starts with context analysis, followed by the dispatch to the appropriate algorithm, the execution of algorithm and an evaluation, which then have a loop back, either to the dispatch of another algorithm or a complete new context analysis.

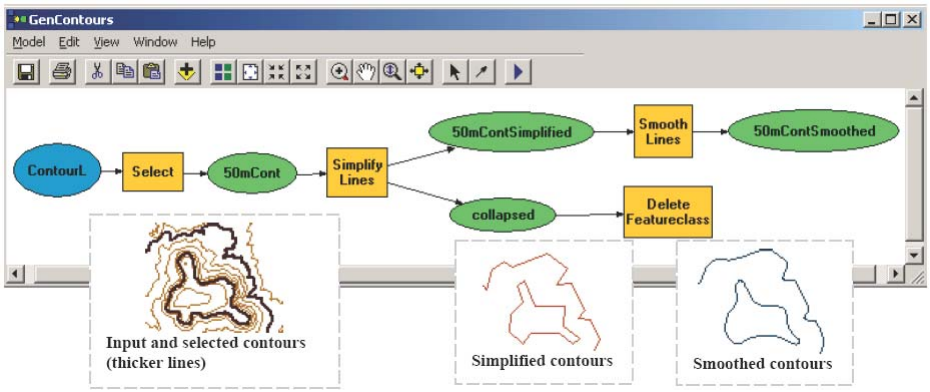

Figure 2. An experimental generalisation sequence in ModelBuilder (Lee, 2003).

Downs and Mackaness (2002) used a similar methodology for the generalisation of geological maps based on an adapted framework of Brassel and Weibel (1988) with five separate phases of structure recognition, process recognition, process modelling, process execution and data display. In the paper

they discussed whether algorithms can be combined into a generic rule-based procedure to enable automated generalisation and found the interdependencies that existed between generalisation operations made it very hard to separate workflows sufficiently.

3. Theory of workflow modelling

Workflows allow the flow and orchestration of processes to be defined. Their expressiveness is comparable to programming languages. In contrast to these, workflows can be defined with a graphical editor and the flow visualised as a diagram. The graphical interface also enables users with no knowledge of programming to create workflows for the control of the generalisation processes. In contrast to batch processing workflows support for instance conditional jumps, loops and concurrent processes (van der Aalst et al., 2003). In this sense workflows close the gap between manual and fully automatic orchestration. A workflow management system consists of a workflow engine and a workflow editor. The workflow engine executes the workflow, which are defined either interactively with the workflow editor or automatically.

3.1 Workflow definition

To allow a more precise and formal definition of workflows Petri nets are often used (van der Aalst, 1996; van der Aalst, 1998; Knorr, 2001). Petri nets (Petri, 1962) are based on bipartite directed graphs and allow a mathematical representation of concurrent flows and distributed systems. So a workflow can be defined as a bipartite directed graph. The graph consists of a set of places and a set of transitions as nodes, which are disjoint. For workflows the transitions represents the tasks (activities) to execute and the places the data. The nodes are connected by edges such that never two nodes of the same set share an edge. In the context of workflows the weights of the edges are one. Places may contain zero or more tokens, which can change during the execution of the net. A transition is enabled, if all its input places contain at least one token. An enabled transition can fire anytime between zero and infinity. It then deletes one token from each input place and produces one token in each output place. Thus the execution of the Petri nets are nondeterministic this means for workflows that transitions can fire anytime and multiple transitions can be enabled at the same time. There exist two special nodes, the start and end node of the graph, which are transitions.

So a workflow based on Petri nets can be formally described as a triple:

$$W := (S, T, F)$$

where W is the workflow, S the set of tasks, T the set of transitions (S and T represent the nodes) and F the set of flow relations (edges) with $F \subseteq (S \times T) \cup (T \times S)$. In contrast to Petri nets marks are not explicitly represented.

A detailed discussion about applying Petri nets for workflows can be found in (van der Aalst, 1998; Knorr, 2001). Usually the data is not explicitly represented in the workflow and so the places are “omitted” and are implicitly merged with the transitions. The *transition* in the model of Petri nets corresponds with the *task* of a workflow. A task is consequently enabled, when each input task provides the required data and the task is then able to fire. This ensures that all previous tasks are completed. Thus, it guarantees the order of execution also if the workflow is concurrent or has conditions.

3.2 Workflow patterns

Workflow patterns are an important part of workflows. These patterns allow besides sequential also concurrent (parallel) and conditional routing. The branching of a workflow, dependent on an evaluation function for example is supported by conditional routing. Conditions can also be used to build loops. There are more patterns available to define AND/XOR-splits and -joins, patterns evaluating structures, cancellation patterns and so on. For a detailed discussion about workflow patterns see (van der Aalst et al., 2003; van der Aalst et al., 2005b).

3.3 Workflows are a framework

Workflows introduced in the previous subsections can be understood as a kind of programming framework – a kind of "reduced" programming language. A simple workflow can be defined with the workflow editor per drag-and-drop. For simple batch jobs like a chain of operations it is sufficient. In general workflows are used for more sophisticated jobs and so further program constructs are needed. If for example the result of one operation is used as a parameter of a later operation variables and assignments are necessary.

With the introduction of variables and assignments the concept of data types must be supported. A mechanism has to ensure that assignments and the parameters of the operators are typed correct. Typical errors are for example an assignment of an `integer` value to a `double` variable or the usage of a `double` value where a `boolean` parameter for an operator is required. Other necessary elements to control flows are choices (splits, branches) by analysing expressions (involving variables), which are equivalent to the "if-then-else" statements of programming languages. In the context of workflows they are often called workflow patterns as introduced in the subsection before. To support choices primitive data types like `integer`, `boolean` etc. must be supported. Loops are another desired control flow statement. To allow an effective usage of loops besides the primitive data types a container data type must be supported. With a container it is possible to calculate and to store in a loop different possible solutions. Consequently in a further loop it is possible to reuse the collected data stepwise. Other data types like features, evaluation constraints, etc. must not be known to the system, because they need not to be interpreted or analysed by the workflow management system.

Böhm and Jacopini (1966) showed that any algorithms could be constructed with sequences, assignments, choices and loops. Later it was shown that choices are not necessary and can be replaced by loops. Programming languages base on these control flows. So the introduced workflow definition in conjunction with the described control flows and assignments enables users to model and describe generalisation flows. This approach is closely related to imperative programming. A different programming paradigm, functional programming, is proposed by Frank and Kuhn (1995) in the field of GIS. The main idea of functional programming is to describe algorithms by functions. Functional programming belongs to the group of declarative programming languages. Based on this programming paradigm their approach uses functions, for the flow control choice and recursion and primitive data types like `integer`, `strings` etc.

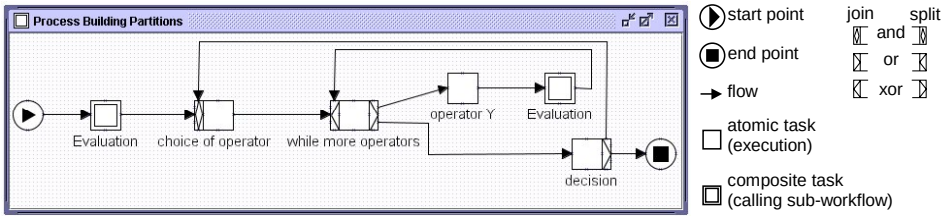

Figure 3. An example of a workflow with start- and end-point, task, choice, loop and call of a sub-workflow.

3.4 Views of a workflow

Workflows come along with a graphical representation of the underlying workflow graph. The graphical representation is restricted to the tasks (nodes of the graph), the flow (edges of the graph) and the end and start point. Choices or loops are a part of the flow and are represented by the visualised edges of the graph but the accompanying conditions are not inevitably visualised. A reason is that conditions evaluate expressions which imply variables and variables are not represented in a visualisation of a workflow. An example of a workflow is shown in Figure 3 – no standardised visualisation exists for workflows and they are often mixed-up with flowcharts. The example in Figure 3 shows an operational view of a workflow with tasks, choices and loops. The example contains a call of a sub-workflow visualised by a double square.

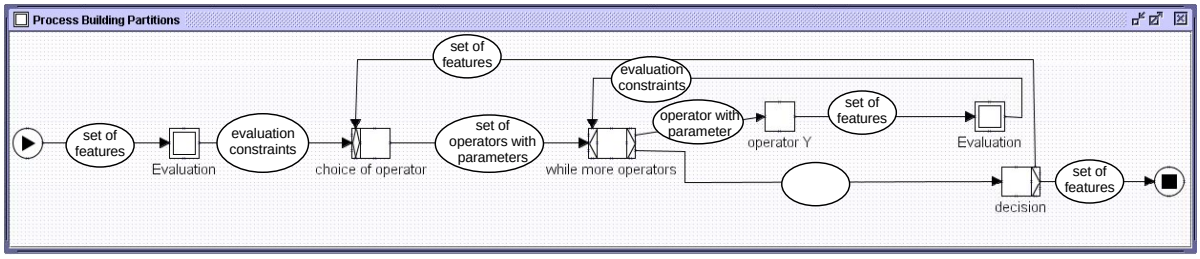

Figure 4. Workflow with output data types

Another possible visualisation could include data types as shown in Figure 4. A more precise description of the data types is done like “set of features” or “set of operators with parameters” instead of only “set”. The data types represent the output of the previous tasks.
If the user can assign interactively variables and parameters the resulting graph is getting much more complex, even if it is restricted to the main variables (Figure 5). Quite often primitive data types as parameters are required like the type `double` for distances and so on, which are not explicitly represented in the figure.
If also relations between features of different resolutions or update relations should be considered the pictorial representation is getting much more complex. A suggestion of relations is visualised in Figure 9. In section 7 the integration of these kinds of relations is discussed.

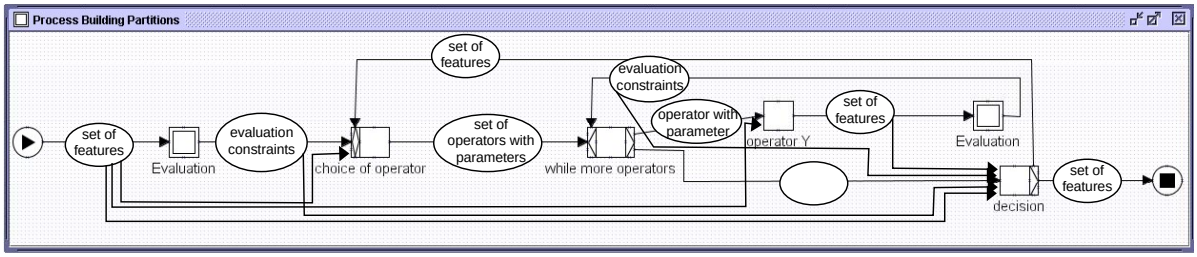

Figure 5. Workflow with data types and flow of data. The integration and visualisation of outputs of a task inside a loop is a little bit confusing and not ambiguous.

4. Concept of a workflow management for generalisation web services

4.1 Web Services for Generalisation

This subsection focuses on the usage of web services for generalisation purposes and their integration in workflows. Detailed introduction and discussions of web services for generalisation can be found in (Neun et al. 2006, Burghardt et al. 2005a). Web services can be differentiated in three groups: support services, operator services and process services (Burghardt et al. 2005a). Support services represent the lowest level of services for generalisation. They can be utilised in generalisation workflows to filter features, enable selections or generate supporting data structure like graphs. Operator services are used for the implementation of generalisation functionality, e.g. simplification, smoothing and collapse. The last category, the process services, enables the control and orchestration of generalisation operators. They realise the situation analysis, conflict detection, choice of operators, evaluation etc. as shown in Figure 6. An example of a processing service will be discussed in detail in section 6.1.
The aim of our work is the design of a workflow management system that supports generalisation tasks but is generic enough that it can also be used for other GIS related tasks. This workflow management system enhances the cartographic GIS *axpand* from Axes Systems (Axes Systems, 2006). Additionally, to the support and integration of the existing functionality of *axpand*, the workflow management system will enable the usage of GIS functions from other providers. Another

objective is the ability to allow the integration and usage of defined workflows through other GIS via web service plug-ins.

GIS tasks are often complex and time consuming especially in the field of generalisation. The application of web services has the advantage that they can be used on distributed systems. Distributed systems support the decentralised and parallel computing, using two or more computers communicating over a network to accomplish independent tasks (Tanenbaum et al., 2002). The workflows based on Petri nets combined with the workflow patterns described in section 3 enable the concurrent execution of flows. Complementary to distributed systems, concurrency describes the property that two or more execution flows are able to run simultaneously (Petri, 1962). The workflow patterns allow the splitting of sequential flows in concurrent ones and the joining of concurrent flows. Thus web services support distributed systems, which suits perfectly to workflows based on Petri nets, with the support of concurrency.

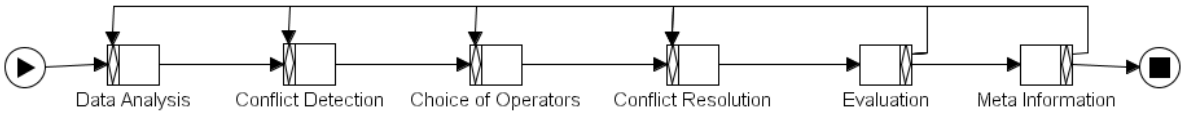

Figure 6. Workflow of the generalisation process (Burghardt et al., 2005b).

4.2 Applying Workflows for Generalisation

Workflows with patterns are a suitable instrument for defining complex generalisations in a convenient way. The user can define a workflow for generalisation by clicking the single generalisation steps together and extend it with conditions, loops or other patterns. A workflow behaves to a user like a generalisation service. Both need input data and parameters, manipulate the data and return a result. There is only a small technical difference. A generalisation service is executed by a service and a workflow by a workflow-engine. But a workflow-engine behaves for a user in the same way as a service: execute the workflow. This leads to the idea of offering workflows as generalisation services.

The functionality of the workflow management systems will be improved stepwise during the implementation. In a first step only workflows with a fixed sequence of tasks will be supported – so-called static workflows. The tasks can consist of generalisation services or “sub”-workflows. Static workflows enclose the expression possibilities of batch-jobs. In a second step, dynamic workflows with conditions and loops will be considered. If a workflow contains for example analyses and evaluation tasks, the workflows can continue dependent on these results. The dynamic workflows are comparable to the agent concept. In a third step, machine learning workflows will be introduced. Therefore, functions will be developed which will generate on the basis of analyse and evaluation functions, workflows appropriate to the situation by machine learning procedures.

5. Architecture of workflow management system

The *axpand* system (Figure 7) is composed of three main components: core *axpand*, a generalisation service architecture and a workflow management system. The generalisation functions of the cartographic GIS *axpand* were extended during the previous project DRIVE (Burghardt et al. 2005b, Bobzien et al. 2006a). The generalisation functions are offered as generalisation services via a so-called generalisation server. A central server, so-called registry server, describes which generalisation service is available and where it can be accessed. Several generalisation servers could exist, offering a variety of generalisation services. The system design considers two network scenarios: intranet and internet. The local autonomous intranet is strictly separated from the internet. The internet scenario allows the access of external generalisation services and the usage of internal services by external applications. Additionally, a workflow management system is integrated. Workflows can be graphically designed by a workflow editor. The editor is provided with all possible generalisation services and workflows by the registry server. All workflows are stored on the registry server. The generalisation server is extended by a workflow engine. So the generalisation server can execute

generalisation services or, if a workflow is called, it can send the workflow to the workflow engine for execution. The engine calls the generalisation server for execution of each task.
For the workflow management system YAWL (Yet Another Workflow Language) is employed, an open source project developed at the Centre for IT Innovation, Queensland University of Technology, Australia (van der Aalst et al., 2005a, http://www.yawl-system.com/). YAWL consists of an editor and an engine and is based on Petri nets extended by workflow patterns. YAWL supports the integration of web services via a WSDL/SOAP interface.

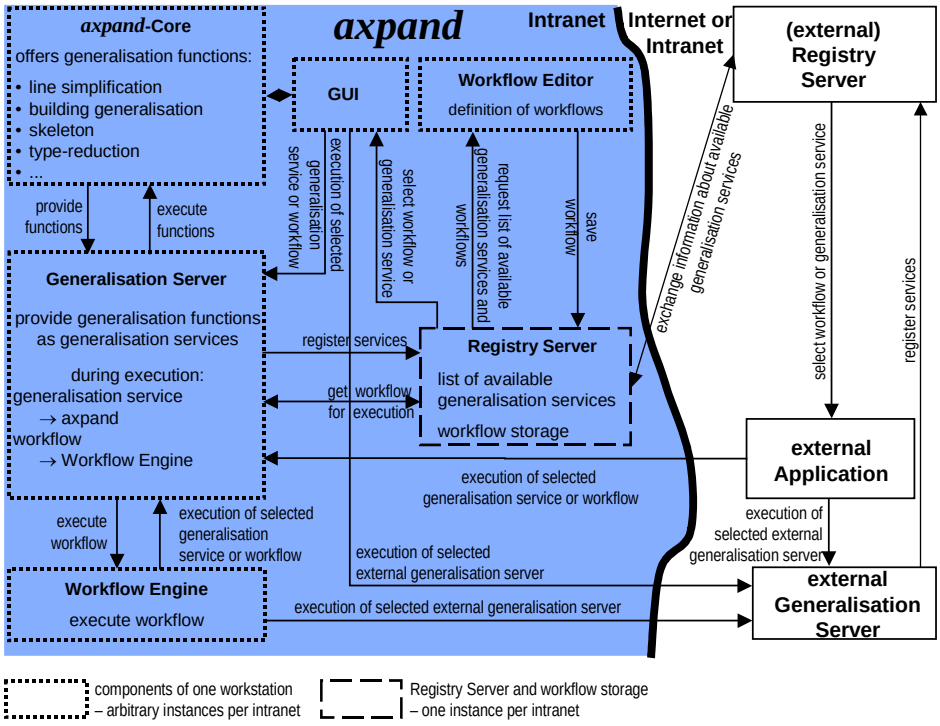

Figure 7. The system consists of the commercial cartographic GIS *axpand* (Axes-Systems, 2006) with its generalisation functionality, the generalisation web service architecture with generalisation and registry server and the workflow management system.

6. Example of a generalisation workflow

6.1 Analyses and evaluation

The generalisation workflow starts in general with a situation analyses and conflict evaluation. The situation analyses contain several pre-processing steps, which results in the construction of intra-resolution relations (Bobzien et al., 2006b). Examples are partonomic relations, neighbourhood relations, semantic relations as well as patterns or structural relations. The amount and combination of possible intra-resolution relations is nearly endless and therefore the user can specify with help of workflow management system, which intra-resolution relations has to be calculated. In the following the construction of partitions as example of partonomic relations will be explained more in detail.

Partitions subdivide the map space in such a way that generalisation tasks can process them independently. An example is the trans-hydro-graph proposed by Timpf (1998) that describes a graph structure derived from transportation and hydrology networks. The trans-hydro-graph can be used to separate the task of building generalisation since buildings must stay inside the faces of the trans-hydro-graph. In the partition example the user can choose the separating feature classes like major and/or minor streets networks, railway networks, river networks or borders of vegetation zones. The result of the explicit calculated partonomic relations is a list of groups, whereby every group can contain any number of buildings. Every intra-resolution relation is associated with an amount of

constraints, which describe an ideal cartographic situation. After situation analysis an evaluation of constraints is carried out to determine conflicts and derive a ranking for the generalisation operators.

Bard (2004) proposed three types of assessment functions to determine the quality of cartographic generalisation. The first type is the characterisation functions, which characterises single or group of features with help of constraints. Second type is the evaluation functions, which compares the states of the features before and after generalisation. The third type contains the aggregation functions, which are used to summarise the individual evaluations. Similar to this methodology, our approach is to calculate a global cost function based on a constraint space (Burghardt and Steiniger, 2005). Features or group of features are placed inside the constraint space depending on their cartographic properties. The distance of the features from the origin is a measure of cartographic conflicts. Features at the origin are not violating any cartographic constraints. The constraints allow the identification of similar cartographic situations for the features, which will be generalised with the same workflow based on a combination of generalisation services. The selection of workflows considers both the degree of constraint violation as well as the importance of constraints. In a first instance the assignment can be static with an interactive component. An evaluation after generalisation can be applied to create a probability of successfully used workflows. With this technique a ranking of different workflows can be derived depending on the position of the features inside the constraint space.
The model of a constraint space is based on the generalisation state diagrams, which were used in the context of agent modelling for generalisation (Ruas, 2000). The usage of parallel plots (Figure 8) has the advantage that constraints for any number of features can be shown as well as constraints for a group of features associated with an intra-resolution relation like the density constraint for a partition. The axes of this n-dimensional space represent n cartographic constraints with their degree of satisfaction. The axes are scaled to the interval between [0, 1], whereby a value greater zero means that the constraint is violated. The constraint values are equivalent to severity from the agent model.

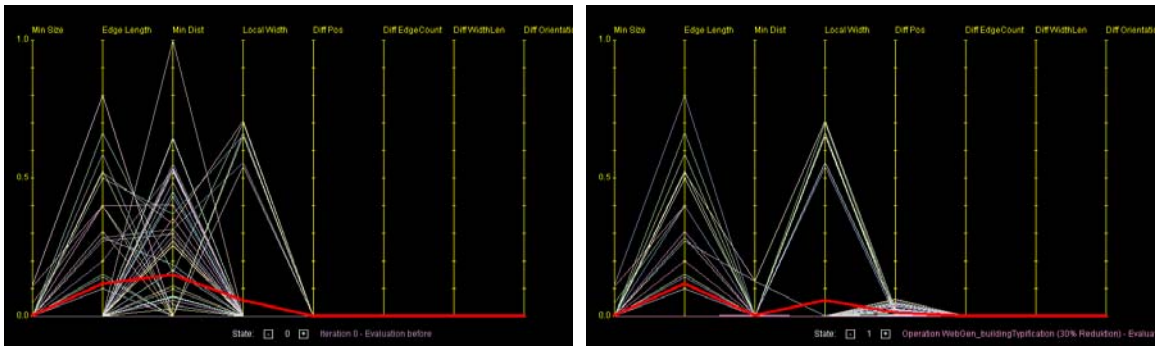

Figure 8. Parallel axis plot for the visualisation of constraint values for any number of features, before (left) and after (right) typification of buildings. Typification service solves the minimal distance constraint (3rd vertical axis) but creates differences in the position constraint of the centroids (5th vertical axis).

6.2 Workflow for building generalisation

This subsection treats an example of a workflow for the generalisation task of building generalisation. The workflow starts with all features derived by a spatial selection. The generalisation task can be divided into two steps: the partitioning of the buildings (pre-processing) and afterwards the building generalisation (generalisation loop). The result of the partitioning is a number of groups with buildings, whereby each group represents one partition. In the generalisation loop different generalisation operators are applied to each partition depending on constraint satisfaction. The associated workflow graph is shown in Figure 9 – the upper graph with windows title *Building Generalisation*. In a following step both tasks the partitioning as well as the building generalisation

have to be refined with sub-workflow similar to procedures in programming languages. A task that calls a sub-workflow is visualised by a double square.

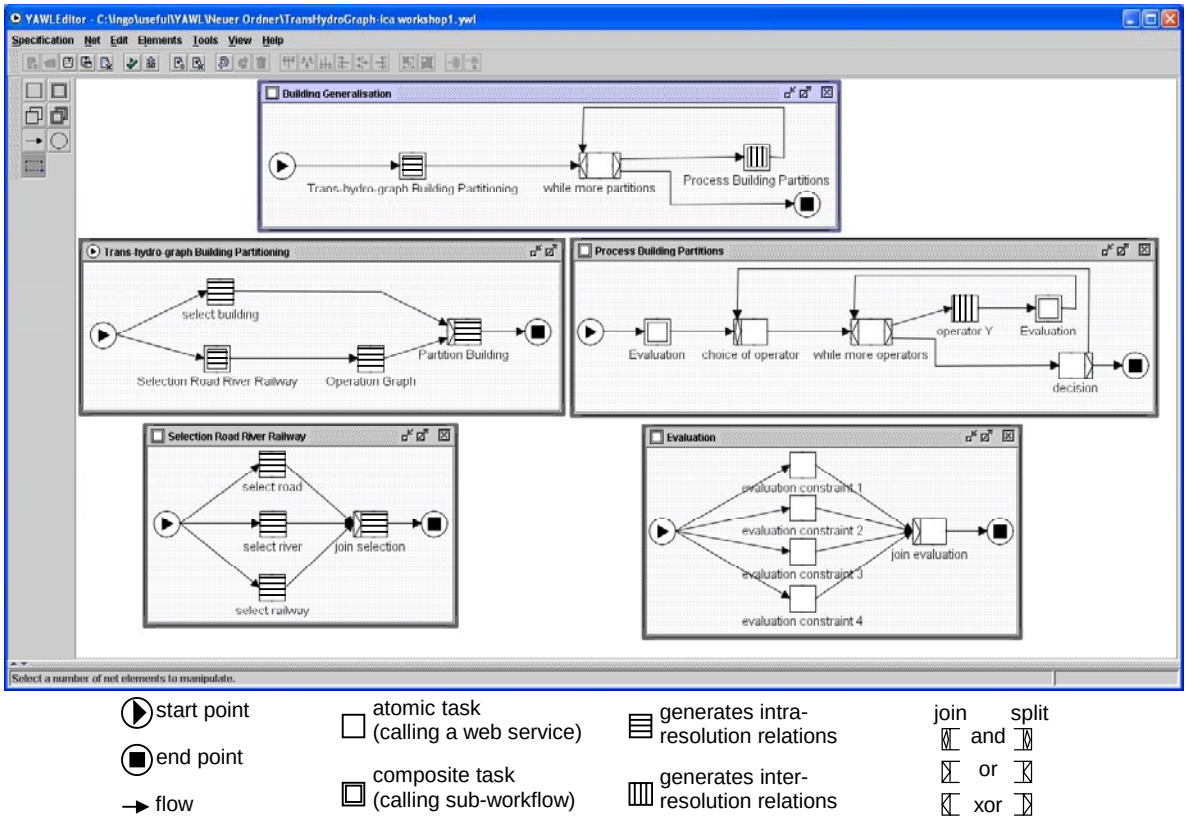

Figure 9. Example of a workflow for building simplification.

For the partitioning of the buildings a trans-hydro-graph is utilised (see subsection 6.1). Therefore three steps has to be carried out, first a semantic selection of linear features and buildings, second the creation of the trans-hydro-graph and finally the association of buildings to the faces of the graph (Figure 9 – workflow graph with the window title *Trans-hydro-graph Building Partitioning*).

The first step of semantic selection can be seen as filter operation to derive the linear features and the buildings from the input features. The user can determine which classes of linear feature will be considered for the creation of the graph. Therefore the sub-workflow *Selection Road River Railway* will be called. The three selection filters (*select road, select river, select railway*) can be applied concurrently to the input set of all features. As shown in the Figure 9 this sub-workflow consists only of atomic tasks. So for example *select road* calls the associated web service for filtering the roads. At the end all linear features are joined.

After semantic selection of the linear features the graph for the partitioning will be generated. Therefore an existing service can be used (*operation graph*). The result is a graph structure consisting of nodes, edges and faces. Only the faces are considered for the association of buildings in the final step in *Trans-hydro-graph Building Partitioning*. The service *partition building* creates groups of buildings representing the partitions, which can be generalised independently. These groups of buildings are an example of an intra-resolution relation as discussed in section 7.

The *while more partitions* construct in the main workflow *Building Generalisation* ensures, that for each partition the sub-workflow *process building partitions* is called. The sub-workflow starts with an evaluation (*Evaluation*), which is needed for the selection of appropriate generalisation operators as well as the comparison of the partition constraints before and after generalisation. The *Evaluation* workflow consists of different constraints describing the state of the building partition (see subsection 6.1). With the selection and weighting of constraints (*evaluation constraint X*) the user can influence

the generalisation process. The results of these evaluations are joined and represent the constraint space (see subsection 6.1).
Based on this characterisation of the building partition, the task *choice of operator* calls a service to determine a ranked list of available operators. The task *while more operators* ensures that the given list of operators are applied (*operator Y*) and the resulting building partition states are evaluated. For the evaluation the same sub-workflow (*Evaluation*) is used as at the beginning, which demonstrates the ability of reusing workflows.
Finally in the task *decision* the best result will be determined. Therefore the derived constraint values before and after the different generalisation operations are compared. The best result will be accepted if the situation has improved. Then the next iteration step tries to apply a further generalisation operation. The loop starts again with the task *choice of operator* and the iteration runs until no further improvement will be reached.

7. Further work - integration of relationships into workflows and generalisation services

An important issue in Geo Information Science is—apart from the handling of features as standalone objects—the handling of *relationships* between features. In the context of generalisation and multi-representation databases (MRDB) exists a classification of three types of relationships between features (Bobzien et al. 2006b):

- *Intra-resolution relations* describe the relationships between features of one resolution at one specific point of time. Examples are partitions, neighbourhood and topological relations.
- *Inter-resolution relations* formulate the relationship between features representing the same geographic entities in different resolutions. These relationship might be of *m*:*n*-cardinality. An example is the representation of five buildings in the source resolution in relation to the representation of the same situation by three buildings within the target resolution.
- *Update relations* model relationships between features that represent the same geographic entities within the same resolution, but at different points of time. Updates usually are one of the three types *inserting*, *deleting* and *modifying*. The latter comprises aggregation as well as parting, thus the update relations might be of cardinality *m*:*n*.

All three types are used in automatic generalisation and automated incremental updating, either implicitly (within the executed functions) or explicitly (as proposed in Bobzien et al. 2006b).
Also in web generalisation services the concept of relationships should be supported. Often these services use internally relations like intra-relations to represent for example neighbourhood. The protocol and data-structures to exchange these information via web services are an active research topic (Neun et al., 2006).

8. Summary

In this paper we introduced and defined the terms workflow and workflow pattern. Afterwards it is shown, that workflows can be utilised to combine web services. Then we described the architecture of a workflow management system with the two main components of workflow engine and workflow editor embedded in a web service environment and axpand. A possible workflow is presented on the example of building generalisation. The detailed explanation focuses on the evaluation services exploiting the concept of constraints. The paper concludes with a discussion on the integration of relationships between features into workflows.

9. Acknowledgements

This research was partly funded by the Swiss Innovation Promotion Agency (KTI/CTI) under the WebGen project (project number 7921.1 ESPP-ES).

References

Axes Systems (2006), *axpand*. http://www.axes-systems.com/e/index.html last visited June 1st 2006.
Bard, S. (2004), Quality Assessment of Cartographic Generalisation. *Transactions in GIS 8 (1)*, pp.

63-81.

Barrault, M., Regnauld, N., Duchene, C., Haire, K., Baeijs, C., Demazeau, Y., Hardy, P., Mackaness, W., Ruas, A., and Weibel, R. (2001), Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization. In: *Proceedings 20th International Cartographic Conference*, Beijing, China, 6-10 August, pp. 2210-2216.

Bobzien, M., Burghardt, D., Petzold, I., and Weibel, R. (2006a), Ableitung von digitalen Vektormodellen – Ergebnisse des Projektes DRIVE. In: *Kartographische Nachrichten* (KN2006).

Bobzien, M., Burghardt, D., Petzold, I., Neun, M., and Weibel, R. (2006b), Multi-Representation Databases with Explicitly Modelled Intra-Resolution, Inter-Resolution and Update Relations, Autocarto In: *Proc. American Congress on Surveying and Mapping*, AutoCarto 2006, Vancouver, USA.

Böhm, C., and Jacopini, G. (1966), Flow diagrams, Turing Machines and Languages with only Two Formation Rules. *Comm. ACM*, 9(5), pp. 366-371, May 1966.

Brassel, K., and Weibel, R. (1988), A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Systems*, 2(3):229–244.

Burghardt, D., Neun, M., and Weibel, R. (2005a), Generalization Services on the Web – A Classification and an Initial Prototype Implementation. *Cartography and Geographic Information Science*, Vol. 32, No. 4, 2005, pp. 257-268.

Burghardt, D., Bobzien, M., Petzold, I., and Weibel, R. (2005b), Cartographic generalisation of large scale maps with axpand. In: *Proceedings of International Symposium on Generalization of Information*, ISGI, Berlin.

Burghardt, D., and Steiniger, S. (2005), Usage of Principal Component Analysis in the Process of Automated Generalisation. In: *Proceedings of 22nd International Cartographic Conference.* La Coruña, Spain.

Downs, T. C., and Mackaness, W. A. (2002), An integrated approach to the generalization of geological maps. *The Cartographic Journal*, Vol. 39, No. 2, pp. 137-152.

ESRI (2006), ArcGIS 9.1, http://www.esri.com/software/arcgis/index.html last visited June 1st 2006.

Frank, A.U., and Kuhn, W. (1995), Specifying Open GIS with Functional Languages. In: *Advances in Spatial Databases*, editors: Egenhofer, M.J., & Herring, J.R., 4th International Symposium, SSD'95 in Portland, ME, Lecture Notes in Computer Science Vol. 951, Springer, pp. 184-195.

Lee, D. (2003), Recent Generalization Development and Road Ahead. *Fifth Workshop on Progress in Automated Map Generalization*, Paris, France.

Li, S., and Coleman, D. J. (2005), Modeling distributed GIS data production workflow. *Computers, Environment and Urban Systems*, 29(4), pp. 401-424.

Hardy, P., and Lee, D. (2005), GIS-based generalization and multiple representation of spatial data. *Proceedings International Symposium on Generalization of Information*, ISGI, Berlin.

Neun, M., Burghardt, D., and Weibel R. (2006), Spatial Structures as Generalisation Support Services. *Joint ISPRS Workshop on Multiple Representation and Interoperability of Spatial Data*, Hannover.

Knorr, K. (2001), Multilevel Security and Information Flow in Petri Net Workflows. In: *Proceedings of the 9th International Conference on Telecommunication Systems - Modeling and Analysis*, Special Session on Security Aspects of Telecommunication Systems, March 2001, Dallas, pp. 9-20

Petri, C.A. (1962), Kommunikation mit Automaten. *Schriften des IIM Nr. 2*, Second Edition, Institut für Instrumentelle Mathematik, Bonn; English translation: Technical Report RADC-TR-65--377, Vol.1, 1966, Griffiss Air Force Base, New York.

Ruas, A. (1999), Modèle de généralisation de données géographiques à base de contraintes et d'autonomie. PhD Thesis, Université de Marne-la-Vallée.

Ruas, A. (2000), The role of meso objects for generalisation. *International Symposium on Spatial Data Handling*, Beijing, China, pp. 3b.50, SDH2000.

Safe Software (2006), FME, http://www.safe.com/products/fme/index.php last visited June 1st 2006.

Tanenbaum, A. S., and van Steen, M. (2002), *Distributed Systems: Principles and Paradigms.* Prentice Hall.

Timpf, S., (1998), Hierarchical structures in map series. PhD Thesis, Technical University Vienna.

van der Aalst, W. M. P. (1996), Three Good Reasons for Using a Petri-net-based Workflow Management System. In: *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*, Cambridge, Massachusetts, pp. 179-201.

van der Aalst, W. M. P. (1998), The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1), pp. 21-66.

van der Aalst, W.M.P., and ter Hofstede, A.H.M. (2005a), YAWL: Yet Another Workflow Language. In: *Information Systems*, 30(4), pp. 245-275.

van der Aalst, W. M. P., ter Hofstede, A.H.M., and Dumas, M. (2005b), Patterns of Process Modeling. In: *Process-Aware Information Systems: Bridging People and Software through Process Technology*, Editors: M. Dumas, van der Aalst, and ter Hofstede, Wiley & Sons, pp. 179-203.

van der Aalst, W. M. P., ter Hofstede, A.H.M., Kiepuszewski, B., and Barros A.P. (2003), Workflow Patterns. In: *Distributed and Parallel Databases*, 14(1), pp. 5–51.