Distributed GML Management with SVG Tools

Keywords: SVG editor, Web mapping, Collaborative work, Geodata, XML

Gunnar Misund
Associate Professor
Østfold University College
Halden
Norway
gunnar.misund@hiof.no
http://www.ia.hiof.no/~gunnarmi/

Biography

> Gunnar is teaching and researching at Østfold University College, Halden, Norway. His main research interests are digital (web) mapping and distributed computing. He is the founder of Project OneMap.

Henning Kristiansen
M.Sc. Student
Østfold University College
Halden
Norway
henning.kristiansen@hiof.no

Biography

> Henning is studying computer science at Østfold University College, Halden, Norway. He is especially interested in game programming and design and implementation of complex user interfaces. He is a member of the Project OneMap Team.

Mats Lindh
M.Sc. Student
Østfold University College
Halden
Norway
mats.s.lindh@hiof.no

Biography

Mats is studying computer science at Østfold University College, Halden, Norway. Among his interests are algorithm design, game programming, web computing and component based systems. He is a member of the Project OneMap Team.

Abstract

We present a browser based GML (Geographic Markup Language) editor implemented with SVG (Scalable Vector Graphics) and DOM (Document Object Model) scripting. The editor is able to treat GML 2.1 compliant data. The geometric editing is based on moving, deleting and inserting points and groups of points. Thematic properties are edited in tables. The editor is designed, within certain limits, to be a multi purpose editor. However, the main target application is data management in Project OneMap. We outline a client/server based peer review submission process using the editor as the main tool.

Table of Contents

1. Background
2. Editor
2.1 Functionality
2.2 Implementation
2.2.1 Window management
3. Application
3.1 Project OneMap
3.1.1 Repository
3.1.2 Gateway
3.1.3 ClearingHouse
3.2 Data Submission by Peer Review
4. Further Work
5. Final Remarks
Footnotes
Acknowledgements
Bibliography

1. Background

The work presented in this paper [1] is part of an ongoing long term effort named Project OneMap [MIS1] . The main objective of this project is to incrementally build a large, distributed global online map. The intention is to provide free access to a repository of geographic information that may be used as base data for a wide variety of applications, such as GIS (Geographic Information Systems) analyses, environmental planning, tourist guides, transportation management and the like. Project OneMap is an open project in every respect, as it is based on Open Source [OSR] and Open Content [OCN] principles, and also adopts a distributed management model. The project is currently hosted by and coordinated from Østfold University College, School of Computer Science. In addition, all major software components are completely platform independent.

Much of the work was carried out as parts of student projects in the Digital Map course at Østfold University College in the spring of 2003 [DM] . The authors of the paper were all connected to the course, as supervisor and students.

The first part of the paper describes the GML editor, and the last part gives a conceptual view of a client/server based peer review process, supporting submissions of new geodata and modifications of existing data in the OneMap system. We close the paper by pointing out directions for future development and presenting some conclusions.

2. Editor

The main rationale for designing and developing the GML editor was to provide a flexible and efficient tool for presenting and editing data represented on the XML (eXtensible Markup Language) format GML, as defined by the OGC (Open GIS Consortium) specification GML 2.1 [GML2] . This version of GML has recently been replaced by the the more complex set of GML 3.0 schemas [GML3] (which are backwards compatible with 2.1). GML 3.0 is expected to be adapted and adopted as an ISO (International Standardization Organization) standard, ISO TC211/19136, hopefully some time during 2004 [ISO] . The main target application is the Clearinghouse part of Project OneMap, as described later in the paper. A fully working demo of a standalone version (without server connection) of the editor is presented in **Figure 1** (See [HLP] help on using the interface).

Figure 1: Working Demo

2.1 Functionality

The GML Editor is designed as a lightweight client application able to run in a standard Web browser, such as the MS Internet Explorer. The server side of the system takes care of translating GML compliant documents into carefully structured SVG instances. After modifying the SVG document with a DOM supporting plug-in (e.g. Adobe SVG Viewer 3.0 **[ADO]**), the changes are transferred back to the server, which applies the modifications to the original GML file. As indicated in **Figure 2** , currently two schemas **[PAC]** are accepted as defining the valid input formats for the editor.

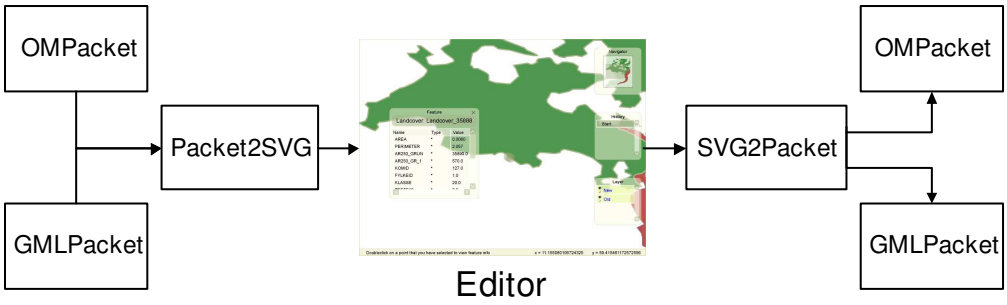

Figure 2: Architecture

A GML document is essentially structuring geodata according to a simple feature approach, as specified in OGC Simple Feature Specification **[SIM]** . A feature is a geographic entity, such as a road, building or lake. A feature is defined by a set of properties. A feature property may fall in one of two categories, either as geometry or thematic (non-geometric) information. The editor is capable of treating modifications of both types of information.

GML geometry is piecewise linear, in the sense that is is based on linear interpolation of point sets of coordinates, e.g. polygons. Hence, the main modifications to such geometries may be decomposed to moving, deleting or inserting a point or set of points. The editor supports both these operations, as illustrated in **Figure 3** .

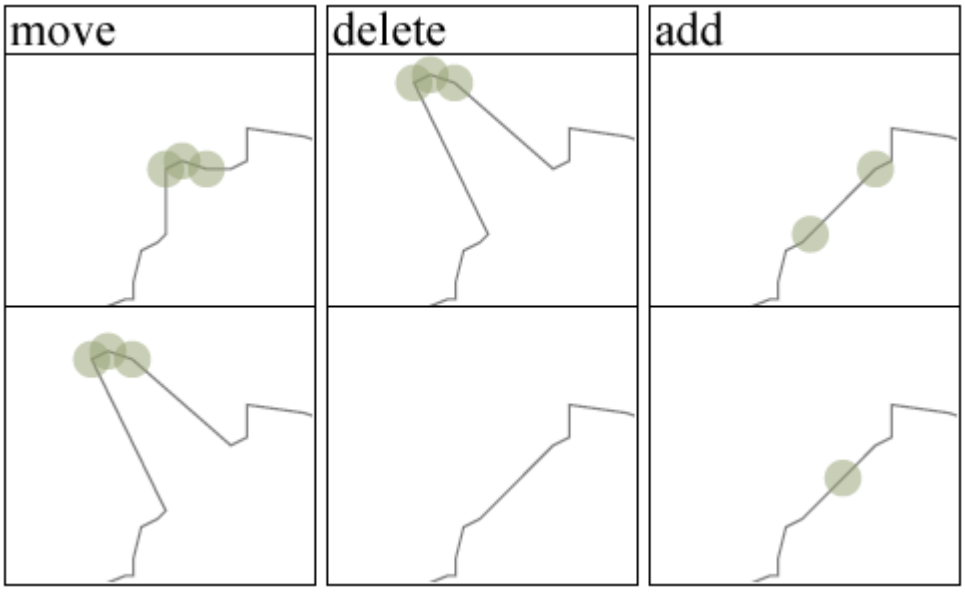

Figure 3: Geometry Editing

The non-geometric information in a GML file can be generalized to a list of an arbitrary number of property-value tuples. More precisely, we are dealing with triples, since the data type (string, integer,

double, etc.) of the property may be specified. The editor is able to modify the values of these properties. By clicking on a the geometry of the feature, a editable property window pops up (see **Figure 4**).

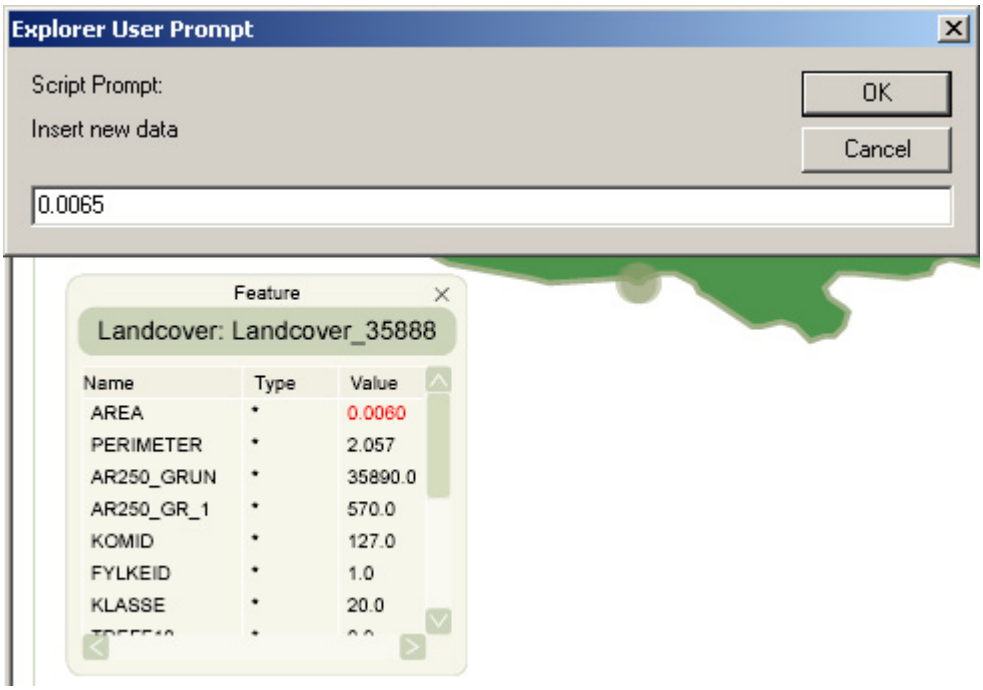

Figure 4: Properties Editing

In the main modus the editor handles a GML file where the modifications and the original data are embedded as separate layers. The editor treats the layers independently, and is able to perform locking and visibility operations on each layer. In **Figure 5** the layers are visible: the new one is locked, and the old is editable. When dragging out the selection rectangle, only points from the old layer are highlighted.

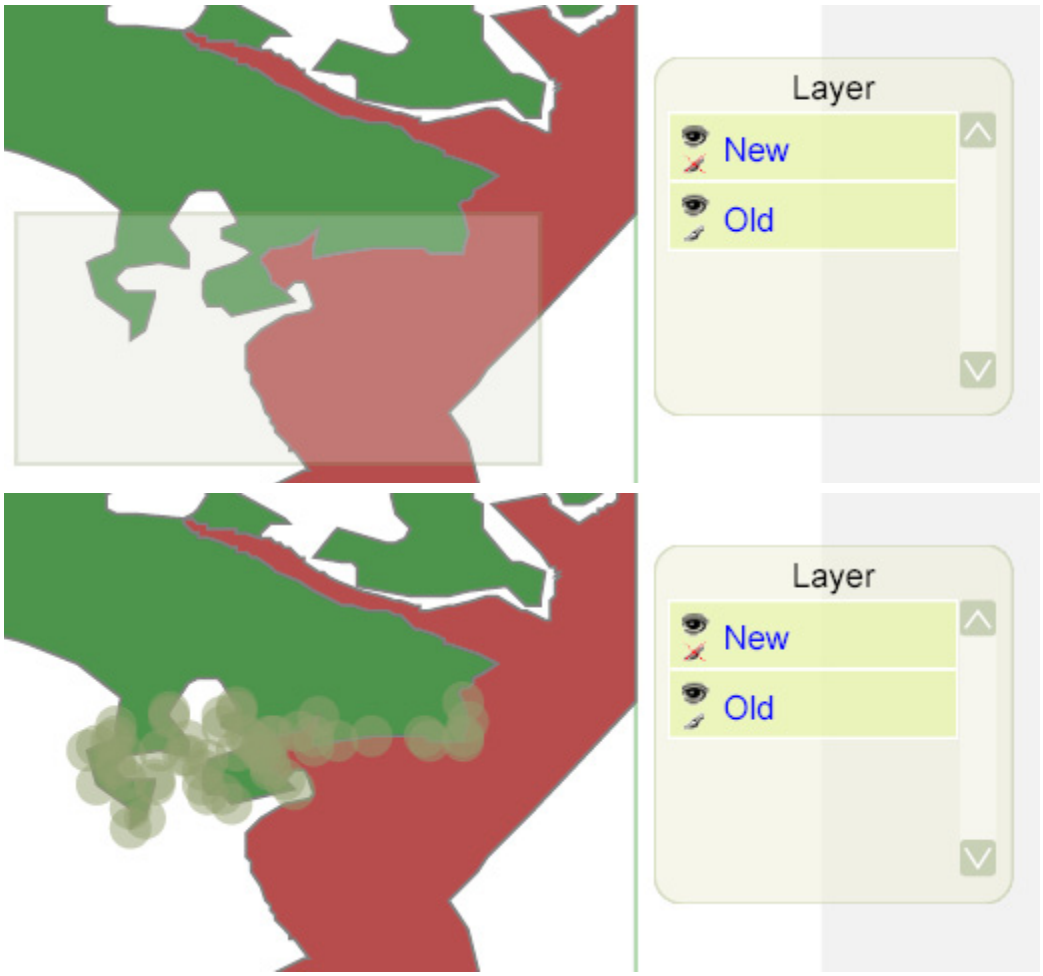

Figure 5: Layer Management

All modifications are recorded in a history list, and it is possibly at any point in the editing process to revert to an arbitrary point in the process and undo/redo the corresponding changes of the document. The history list is scrollable, and each event is labeled with the type of change, as illustrated in **Figure 6** .

Figure 6: History Window

To facilitate zooming and panning, a navigator window has been added, displaying the current area of the main window related to the original view. Zooming may only be performed in the main window (both by zoom rectangle and mouse click). Panning may be done either in the main window (the standard panning tool) or by dragging the location rectangle in the navigator window (**Figure 7**).

Figure 7: Navigator Window

Finally, the editor is fully resizeable in order to support different combinations of display sizes and resolutions combined with the user's preferences. Some examples are shown in **Figure 8** . Note that the resizing takes proper care of relative positions and sizes of the menu windows.

Figure 8: Resizing

2.2 Implementation

The editor has been realized by taking full advantage of the SVG DOM, which is a slight extension of the DOM level 2 interface defined by the W3C (World Wide Web Consortium) **[DOM2]** . Interaction with the DOM interfaces are implemented with ECMAscripts (java) **[ECM]** . The editor is based on inspiration and solutions from a growing number of SVG based web mapping solutions, such as the Vienna project **[VIE]** and the Cleopatra scripts **[CLE]** , and general purpose SVG authoring tools, e.g. the SVG editor from Peto **[PET]** .

One of the design goals was to make the editor reasonably independent of both the browser and the plug-in. This is not easily achieved due to differences in browser support for ECMAscripting and differences in the plug-in implementations of the SVG DOM. However, the editor currently runs on the most common combinations of browsers and plug-ins.

During implementation of the editor, several challenges surfaced. We briefly outline some of the encountered problems and their solutions.

2.2.1 Window management

The editor is designed according to the widespread model with a main working window and several smaller widget windows to facilitate different operations. Some of the windows are permanent, and some are of pop-up/closable kind. All the auxiliary windows are dragable and transparent.

The movable windows are implemented as *SVG in SVG* components with their own coordinate systems. This reduces the complexity of performing various operations within a auxiliary window, independent of it's parent SVG window, and also makes dragging easier to implement, as illustrated in **Figure 8** .

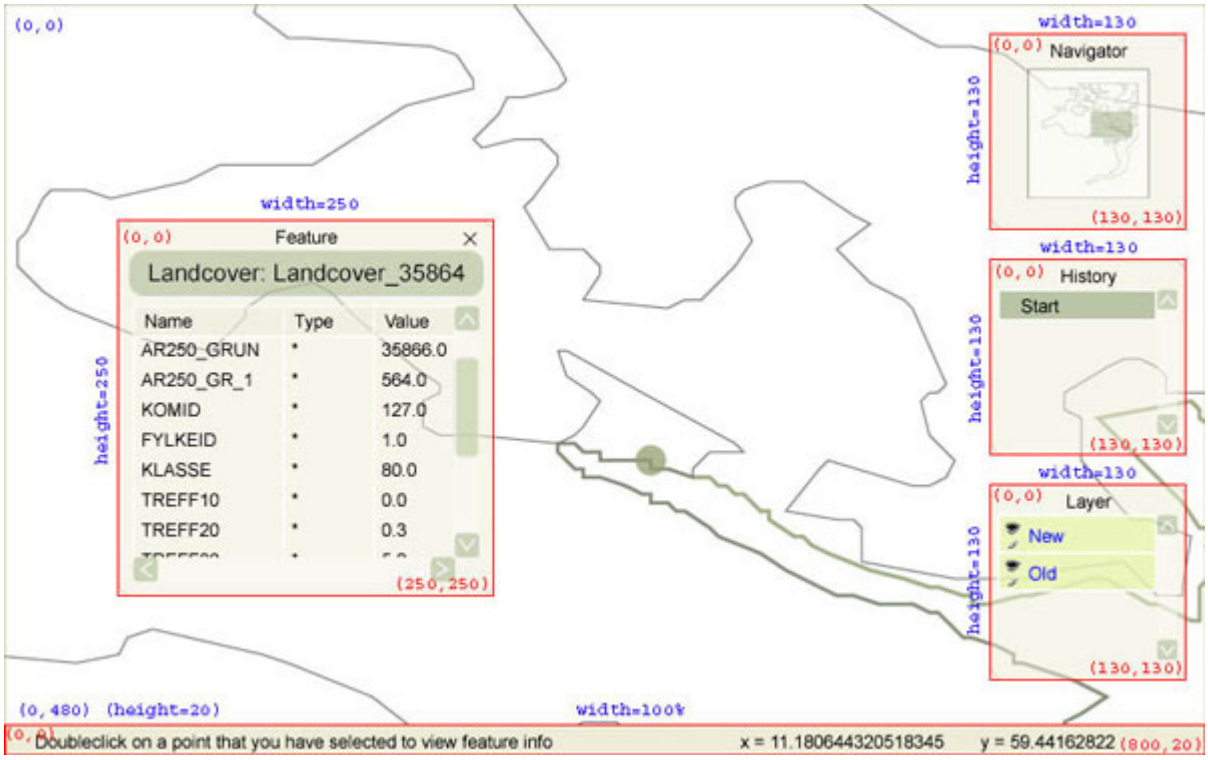

Figure 9: Coordinate Systems in Main and Auxiliary Windows

Figure 9 illustrates the set of SVG windows and their corresponding coordinate systems. The red rectangles represent the SVG windows for the widget windows. All of the graphical objects inside each window are defined in the corresponding coordinate system, indicated by the red text. The main window coordinates are in blue. When resizing the main window, the coordinates for each widget do not change. However, the position of the upper right corner and the length and width will be scaled accordingly.

All windows share a common set of JavaScript functions **[JSC]** , briefly described in the **Table 1** .

Function	Description
init	Initiates the menu and points to elements in the SVG DOM
setMenu	Usually called in the init method and calculates the values needed for that window
moveMenu	Moves the menu and keeps track of the current position
scaleMenu	Scales the menu according to the resized main window
startClickMenu	Called on a mouse-over-window event and prepares potential actions
stopClickMenu	Called on mouse-up-inside-window and initiates proper actions

Table 1

What the different functions actually do is specific to the menu item it is applied to. This is however an outline of the basic steps performed on every mouse event in the windows.

3. Application

The editor may be used in a variety of applications, but some of the features have been implemented with a specific application in mind. We give a brief outline of this application, the Clearinghouse component of Project OneMap.

3.1 Project OneMap

OneMap is functionally comprised by three main components: *Repository*, *Gateway* and *Clearinghouse*. The alpha version has been up and running for a year, and the beta version will be released later this year.

3.1.1 Repository

The Repository is a distributed storage infrastructure. This is basically a (huge) set of GML files

structured to efficiently support retrieval and updating of the geodata comprising the world map. The total amount of base data is currently in the magnitude of 50 M points, and is expected to expand to around 500 M points within the end of the year. The data is structured in levels of detail to facilitate global overviews as well as street level inspections. The files are distributed redundantly on a set of servers. This facilitates both storage scaleability and parallel processing of retrieval queries and updating procedures.

Currently we use a straight forward directory based management of the distributed data. However, work is in progress to implement a peer-2-peer infrastructure to avoid bottlenecks and single-points-of-failure. The communication within the OneMap infrastructure is based on Web Services paradigms and technologies. The main interface to the repository is defined as an OGC WFS (Web Feature Service) **[WFS]** implemented with a (slightly modified) open source framework from Deegree **[DEG]** . The services are based on Java Servlet technology.

3.1.2 Gateway

The Gateway is a browser based user interface for retrieval of OneMap data. We are currently using an SVG/Javascript implementation, which provides simple but sufficient navigation and query possibilities. In addition to the SVG view, the users may download the GML formatted version of the response. The GateWay is based on the same principles as the GML Editor, and is presented in more detail in **[MIS2]** .

3.1.3 ClearingHouse

One of the main objectives of OneMap is to make it possible to build a huge map in an incremental and uncoordinated manner with contributions from a wide variety of parties, but still guarantee a reasonable level of reliability and quality. The main problem is to integrate submissions with the existing geodata. We address the problem by using a framework based on the well known principles of *peer review*. The process of contributing new geodata or updating existing data is outlined in the next section.

3.2 Data Submission by Peer Review

Each submission to the OneMap Repository is "owned" by the contributing person or party. The submission process supports a bottom-up quality control. It is built upon the assumption that the owners of already submitted data are the most competent to assess the quality of data submitted in the vicinity of their own data.

There are two main user roles in the submission process, the submitter and the peers. Note that the following description is based on geodata submission. However, a similar process covers

management of the OneMap Featuretype Catalogue. This is essentially a database of the existing featuretypes, say river, highway, school etc., which may be modified by the users in a similar way as the geodata. The featuretype management falls outside the scope of this paper.

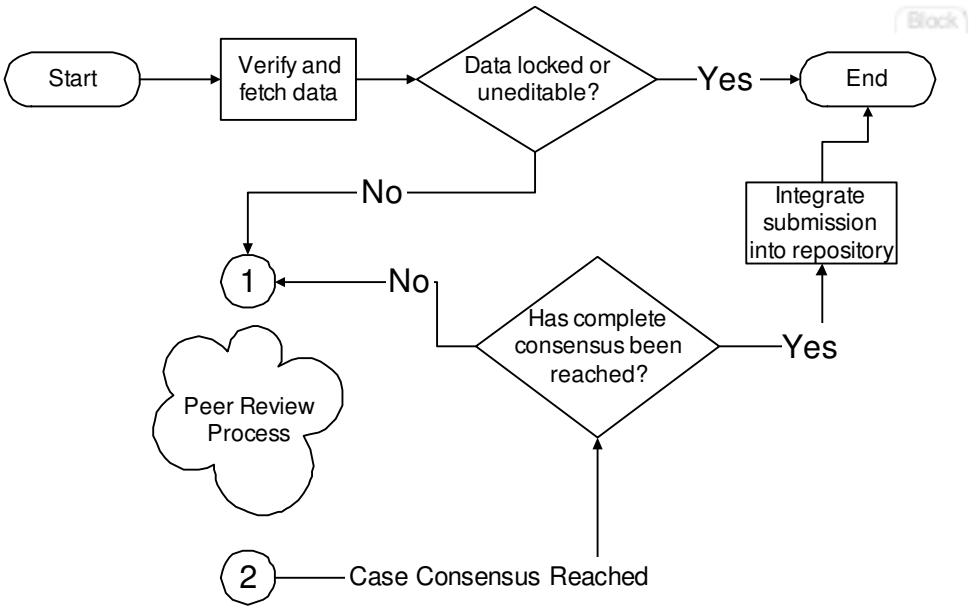

Figure 10: Submission Process

The overall geodata submission process is outlined in **Figure 10** . The submitter uses the ClearingHouse browser based interface and uploads a submission. A submission may be an addition to the data repository, i.e. in the form of new feature instances, or it may define changes, such as updates and refinements of existing data. The submission must be represented as a GML file conforming to the relevant XML schemas **[PAC]** . The ClearingHouse server then identifies a set of peers, i.e. owners of data that is affected by the submission. Then an interactive peer review process is initiated and is terminated when consensus is reached. Finally, the submission is integrated in the OneMap Repository.

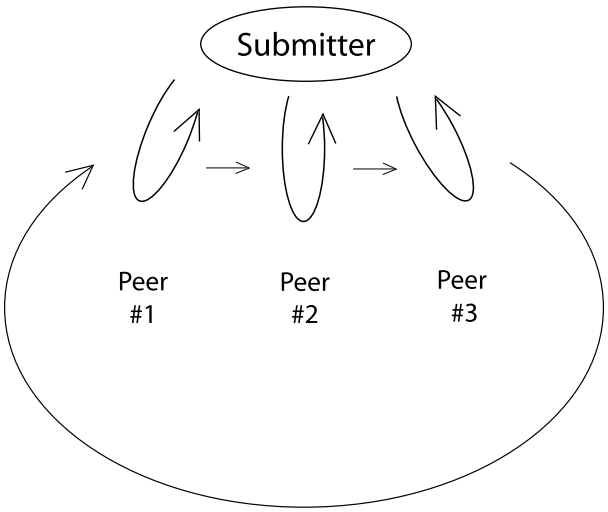

Figure 11: Main Review Cycle

The peer review process consists of a number of main iteration cycles **Figure 11** . Each main cycle starts with a review process between the submitter and the first peer in the peer group list. Suggestions of modifications of the submission is passed back and forth until the submitter and peer both are satisfied **Figure 12** . Then the (possibly) modified submission is subject to a review between the submitter and the second peer, ... and on it goes until the complete peer group finds that the submission is OK. The criterion for consensus may vary depending on the type of data submitted, from complete majority to some partial agreement.

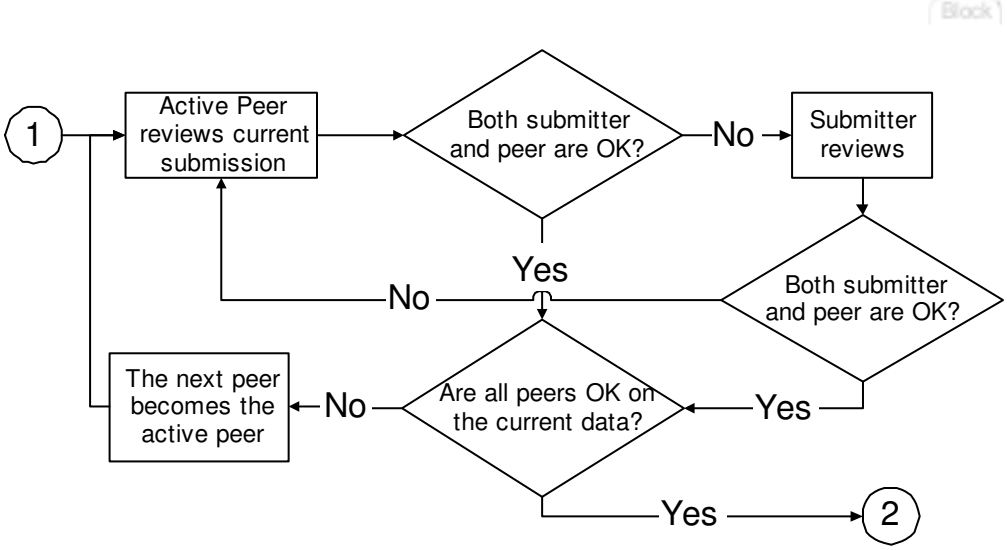

Figure 12: Submitter - Peer Process

The main tool in the review process is the *integration maps*. These maps are presented in the GML Editor, and focus on regions with zero or more conflicts or inconsistencies between the submitted and existing data. **Figure 13** shows how a submission has been broken down in two integration maps, which both are subject to their own review processes. The review process is terminated when all integration maps have been agreed upon.

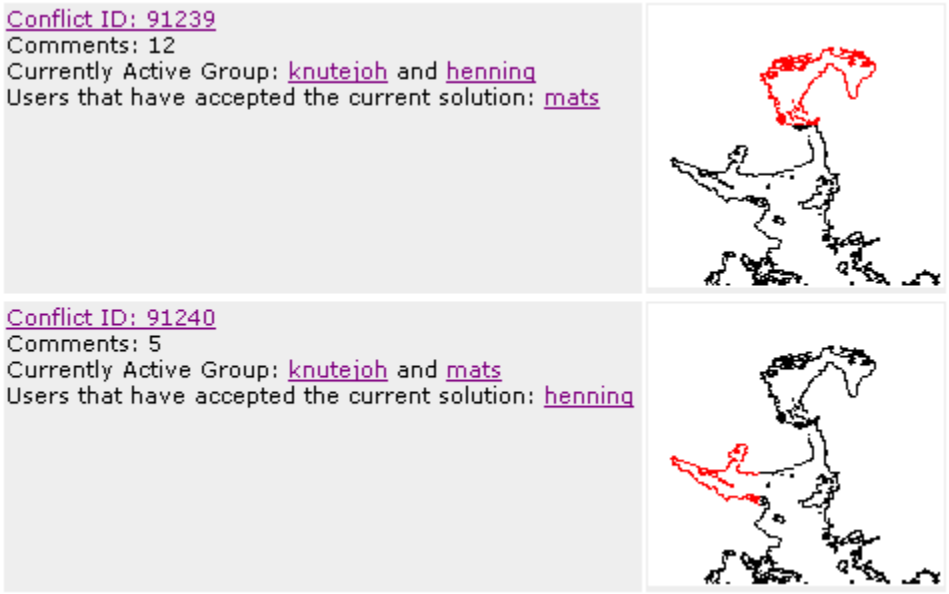

Figure 13: Integration Maps

4. Further Work

Three main extensions are planned. First we would like to make the editor more robust with regard to the kind of GML documents it is able to treat. We are in the process of designing a method that will remove the step where GML files have to be transformed to a predefined packet format. This method will map the GML structure more or less in a 1-to-1 manner to a corresponding SVG document and back again.

If the above approach proves successful, we will be able (with minor adjustments) to also treat GML 3 documents.

The second extension will make it possible to interactively build completely new features, including both geometric and thematic information.

5. Final Remarks

The development of the GML Editor has proved that lightweight client side browsers and editors may very well be implemented with the combination of SVG plug-ins and scripting.

The open nature of this programming approach makes it easy to learn, borrow and modify from a wide variety of existing applications. In addition, the XML based development environment makes it possible to choose from a large number of low cost (or free) and high quality tools. All in all, relatively small efforts were put into the project. The main bulk of the implementation has been carried out by a team of students, in short time, with no or little prior knowledge of SVG, digital mapping or even XML technologies, but with a solid foundation in traditional computer science and programming in general.

Footnotes

1. This paper is part of the OneMap Documentation Series (OMD 00303). The OneMap publications serve two purposes: 1) To disseminate knowledge and experiences that might be useful for other parties and applications, and 2) To encourage feedback and discussion in a broad setting. The presented results are *not* to be regarded as "final solutions", rather as a starting points for further discussions and developments.

Acknowledgements

The author is grateful for the financial support offered by Østfold University College, which made it possible to finalize and refine results from the Digital Maps course. It also made it possible for part of the OneMap Team to attend the SvgOpen conference.

Bibliography

[ADO]

Adobe SVG Viewer 3.0. Adobe Systems Incorporated. http://www.adobe.com/svg/overview/whatsnew.html.

[CLE]

Cleopatra GIS Data Plug-in. SchemaSoft, Vancouver, Canada http://www.schemasoft.com/cleopatra/index.shtml.

[DEG]

deegree 1.0.11. Project Deegree. http://deegree.sourceforge.net/index.html.

[DM]

Digital Maps. Gunnar Misund, Østfold University College http://www.ia.hiof.no/digmap/.

[DOM2]

Document Object Model (DOM) Level 2 Specification. W3C Recommendation 13 November, 2000 http://www.w3.org/TR/DOM-Level-2-Core/.

[ECM]

ECMAScript Language Specification, Standard ECMA-262. ECMA International Standard, 3rd edition (December 1999) http://www.ecma-international.org/publications/files/ecma-st/Ecma-262.pdf.

[GML2]

OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.2 . Open GIS Consortium, Inc., 2003-17-09 (OpenGIS Project Document Number 02-069) http://www.opengis.net/gml/02-069/GML2-12.html.

[GML3]

OpenGIS® Geography Markup Language (GML) Implementation Specification, version 3.0. Open GIS Consortium, Inc., 2003-01-29 (OGC 02-023r4) http://www.opengis.org/techno/documents/02-023r4.pdf.

[HLP]

GML Editor Help. Henning Kristiansen and Mats S. Lindh help.html.

[ISO]

Geographic information - Geography Markup Language. International Organization for Standardization http://www.isotc211.org/scope.htm#19136.

[JSC]

GML Editor Java scripts. http://www.ia.hiof.no/~gunnarmi/svgopen03/editor_script_files.zip.

[MIS1]

The One Map Project. Gunnar Misund and Knut-Erik Johnsen. In online proceedings from GML Dev Days, Vancouver, July 2002 http://www.gmldev.org/GMLDev2002/presentations/MakingMapsfromGML/johnsen/one_map_misund_johnsen.html.

[MIS2]

The OneMap Gateway - An SVG/WFS Interface To Distributed Geodata. Gunnar Misund, Henning Kristiansen, Mats Lindh, Knut-Erik Johnsen. Student poster, SVG Open 2003 Conference and Exhibition, Vancouver, Canada, July 13-18, 2003 http://www.svgopen.org/2003/paperAbstracts/TheOneMapGatewayAnSvgWfsInterface.html.

[OCN]

OpenContent License (OPL). OpenContent, Version 1.0, July 14, 1998. http://opencontent.org/opl.shtml.

[OSR]

The Open Source Definition. Open Source Initiative (OSI), Version 1.9 http://www.opensource.org/docs/definition.php.

[PAC]

GML Editor XML schemas. http://www.ia.hiof.no/~gunnarmi/svgopen03/editor_scemas.zip.

[PET]

SVG Editor. Christopher B. Peto http://resource-solutions.de/svgeditor/.

[SIM]

OpenGIS Simple Features Specification For SQL Revision 1.1. Open GIS Consortium, Inc.,OpenGIS Project Document 99-049, May 5, 1999 http://www.opengis.org/techno/specs/99-049.pdf.

[SVG]

Scalable Vector Graphics (SVG) 1.1 Specification. W3C Recommendation 14 January 2003 http://www.w3.org/TR/SVG11/.

[VIE]

Vienna - Social Patterns and Structures. Andreas Neumann, Version 1.1 (12.3.2000) http://www.karto.ethz.ch/neumann/cartography/vienna/.

[WFS]

Web Feature Service Implementation Specification. Open GIS Consortium, Inc., 19-September-2002 (OGC 02-058) http://www.opengis.org/techno/specs/02-058.pdf/.