Diss. ETH No. 19824

Cartographic Web Services

IONUȚ IOSIFESCU-ENESCU

2011

This page is intentionally left blank

DISS. ETH NO. 19824

Cartographic Web Services

A dissertation submitted to

ETH ZURICH

for the degree of
Doctor of Sciences

presented by
IONUȚ IOSIFESCU ENESCU
Diploma in Geodesy, Technical University of Civil Engineering Bucharest
Master of Science ETH in Computer Science
born December 20, 1979
citizen of Romania

accepted on the recommendation of
Prof. Dr. Lorenz Hurni, examiner
Prof. Dr. Lutz Plümer, co-examiner
Dr. Hans-Rudolf Bär, co-examiner

2011

This page is intentionally left blank

“The best way to predict the future is to invent it. […] You can be proactive about the future; you don't have to be reactive. The whole idea of having scientists and technology is that those things you can envision and describe can actually be built.”

Alan C. Kay

[http://www.ecotopia.com/webpress/futures.htm]

This thesis is dedicated to my dear wife Cristina.

Acknowledgements

I am very grateful for everything that I have learned during my doctoral studies from numerous outstanding people.

First and foremost I am grateful to my advisor Prof. Dr. Lorenz Hurni for his mentorship during the doctoral work. I thank him for generously supporting my personal growth throughout the years of my PhD studies. He has certainly shaped me as a person with his personal example of how to be responsible and dedicated.

I also want to thank Prof. Dr. Lutz Plümer and Dr. Hansruedi Bär for kindly co-supervising this thesis. I am especially grateful to Dr. Bär for his valuable support during the writing of this thesis. With his advices and corrections, he helped me through this last difficult step of the doctoral work.

I am very grateful to Dr. Marco Hugentobler for showing me how to implement ideas in software. He helped shape my research by introducing me to QGIS and by implementing my theoretical ideas in working QGIS mapserver code.

I would also like to thank Sandra Zeder, Andreas Eberle and Felix Ortner, as they had the courage to entrust the outcome of their student works on emerging concepts and software. Furthermore, I am grateful to all my colleagues from the Institute of Cartography and Geoinformation at ETH Zurich for surrounding me with a pleasant and stimulating working environment.

Finally, I am indebted to my wife Cristina for her loving support throughout all these years.

Zurich, 01.09.2011

Abstract

The traditional cartographic workflow is highly dependent on manual or semiautomatic data selection and symbolization. In an era where computer automation is part of our everyday life, the issue of automatically creating high quality maps from GIS content is still very challenging and time-consuming, and this in spite of the power of Web and database technologies. Moreover, many decisions must be taken on short notice based on cartographic visualizations, for instance in the field of risk management.

The above considerations shape the motivation of this thesis. Inspired from the current developments in the field of Cartography and GIS, electronic atlases and map representation definitions such as CartoML, DiaML and Symbology Encoding (SE), this work introduces the generic concept of cartographic Web services as a method for Web Cartography. The thesis proposes the Cartographic Web Services (CartoWS) as a solution for automatically producing maps from geodata based on a generic and expressive map representation. The maps are obtained from GIS data as geo-referenced output images via a Web-based interface.

As previously mentioned, the concept of cartographic web services relies on the map representation. This work defines a map representation adapted for the needs of cartography. The map representation is a formal description of the map that coordinates all aspects of transforming the GIS data into the map image, including geographical bounds, scale selection and, most important, the symbolization. It is based on the OGC standards Styled Layer Descriptor (SLD) and Symbology Encoding (SE). Generic symbols, diagrams, filtering or masking rules are also defined to be compatible with current OGC standards. Moreover, the inclusion of GIS functionalities such as intersection, union and difference in the map representation process allows for a precise differentiation in the symbolization of features, that otherwise would be possible only by preprocessing the data.

The generic concept of CartoWS is specified by the Map and Diagram Service Interface (MDSI). Making use of specific computer science knowledge, such as Web services and their mechanism, concepts and properties, it is specified the interface that details the operations of CartoWS. The definition of the MDSI follows and enhances the Web Map Service (WMS) standard. The QGIS map server is the software implementation of the MDSI. Furthermore, the QGIS map server demonstrates the utility of CartoWS by proof-of-concept applications.

The proposed Cartographic Web Services offer functionalities that support the development of cartographic applications on the Web. A Service Oriented Architecture (SOA), like the one recommended for the CartoWS usage, is the basis for many competitive internet-ready applications, specifically for the creation of dynamic Web resources. On the basis proposed in this thesis, future research may consider a closer cooperation between Cartography and GIS, especially regarding the development of Web-based cartographic applications and Web GIS.

Kurzfassung

Der traditionelle kartografische Arbeitsablauf stützt sich stark auf die manuelle oder halbautomatische Datenselektion und Symbolisierung. Die Computerautomatisierung prägt heute bereits viele Bereiche unseres täglichen Lebens. Trotz der Leistungsfähigkeit der Web- und Datenbanktechnologien ist die Ausgabe von automatisch erstellten, qualitativ hochwertigen Karten aus GIS-Daten dennoch sehr herausfordernd und zeitintensiv. Zudem müssen viele Entscheidungen, welche auf kartografischen Visualisierungen basieren, kurzfristig getroffen werden wie zum Beispiel im Bereich des Risikomanagements.

Die obigen Überlegungen dienen zur Motivation dieser Doktorarbeit. Inspiriert von den derzeitigen Entwicklungen im Bereich von Kartografie, GIS, elektronischen Atlanten und Definitionen für Kartendarstellungen wie zum Beispiel CartoML, DiaML und Symbology Encoding, stellt diese Arbeit ein allgemeines Konzept der kartografischen Webdienste als eine Methode der Webkartografie vor. Als Lösungsansatz für automatisch aufbereitete Karten aus Geodaten werden die Cartographic Web Services (CartoWS) vorgeschlagen, welche auf einer allgemeinen und aussagekräftigen Kartendarstellung basieren. Die Karten werden als georeferenzierte Bilder aus GIS-Daten mittels Web-basierten Schnittstellen bezogen.

Wie oben bereits erwähnt, beruht das Konzept der kartografischen Webdienste auf der Kartenrepresentation, und die vorliegende Arbeit definiert eine auf die Bedürfnisse der Kartografie abgestimmte Darstellungsweise. Diese Representation stellt eine formelle Beschreibung der Karte dar, welche alle Aspekte der Transformation von GIS-Daten in das Kartenbild umfasst, einschliesslich geografischer Grenzen, Massstabsauswahl und Symbolisierung. Die Kartendarstellung basiert auf den OGC Standards Styled Layer Descriptor und Symbology Encoding. Allgemeine Symbole, Diagramme, Filter- oder Maskierungsregeln werden ebenfalls definiert, um eine Kompatibilität mit den aktuellen OGC Standards zu gewährleisten. Darüber hinaus erlaubt die Einbeziehung von GIS-Funktionalitäten wie zum Beispiel Überschneidung, Vereinigung und Differenz, eine präzise Differenzierung in der Symbolisierung von Features, welche sonst nur in der Vorverarbeitung der Daten möglich gewesen wäre.

Das allgemeine Konzept der CartoWS wird durch das Map and Diagram Service Interface (MDSI) spezifiziert. Durch die Nutzung von spezifischen Computertechnologien, wie beispielsweise Webdienste und ihre Mechanismen, Konzepte und Charakteristiken, wird eine Schnittstelle spezifiziert, welche näher auf die Methoden der CartoWS eingeht. Die Definition der MDSI-Schnittstelle berücksichtigt und unterstützt den Web Map Service (WMS) Standard. Der QGIS Map Server ist eine Softwareimplementation der MDSI-Schnittstelle und zeigt den Nutzen der CartoWS durch Proof-of-Concept-Anwendungen.

Die vorgeschlagenen Cartographic Web Services bieten Funktionalitäten an, welche die Entwicklung von kartografischen Anwendungen im Web unterstützen. Eine Service Oriented

Architecture (SOA), ähnlich der für die Verwendung der CartoWS vorgeschlagen, bildet die Basis für viele internetfähige Anwendungen, besonders für die Erzeugung dynamischer Web-Inhalte. Anhand der in der Doktorarbeit aufgeführten Ergebnisse sollte die zukünftige Forschung eine engere Kooperation zwischen Kartografie und GIS berücksichtigen, besonders was die Entwicklung von Web-basierten kartografischen Anwendungen und Web-GIS betrifft.

Contents

1 Introduction and Motivation

This chapter introduces the context of the research, its motivation and the driving factors. The focus lies on finding a solution towards an automatic generation of maps from GIS data by means of cartographic Web services. After stating the problem, the context of the research asserts its intrinsic relation with two European projects and the relevant practical requirements that shaped the research questions and objectives of this thesis.

1.1 Personal Motivation

Creating a map always poses many challenges, and cartographic experience is required for finding solutions to the particularities of every new map product. This was the main insight that I developed about cartography in my one-year research scholarship at ETH Zurich in the academic year 2004 – 2005.

During this research year, I was interested in creating interactive maps for the Web based on data managed by Geographic Information Systems (GIS). I had the opportunity to learn various technologies for Web Cartography and consequently I created two prototype applications making use of the Internet graphic standard SVG (Scalable Vector Graphics) and the PostgreSQL database. The first Web mapping application was a “Learning platform for hazard analysis and visualization” with the purpose to introduce the GIS-based approach to the mapping of natural hazard susceptible areas [Iosifescu et al., 2006]. The second application was a “Visual Metadata Search Tool” for the browsing of the metadata repository of the Research Network on Natural Hazards at ETH Zurich [Hurni et al., 2006a].

During the development of these Web applications I have discovered that creating the map content from GIS data is still very challenging and time-consuming in spite of the power of Web and database technologies. On one hand, I felt the need for different concepts, methods and technologies that would improve the workflow for creating interactive Web maps. On the other hand, there was the important issue of the reusability of the created Web mapping tools and cartographic knowledge invested in creating the map content. Therefore, I became personally motivated to understand why it was not possible to have a simpler and clearer workflow for using GIS data for Web Cartography. During my work on this thesis, I understood that cartographic Web applications were previously developed using the newest technologies, but without a generic architecture and without well-defined, reusable concepts. Instead of developing a new mapping application from the scratch every time, it made sense to me that it should be possible to create the map content automatically or with little additional effort once similar applications were already created. In addition, during the time I was starting to formulate the research question, I became aware of requirements coming from the environmental sciences. The Institute of Cartography has an impressive experience in cartography for environmental information systems [Hurni et al., 2006]. These requirements and experiences helped to shape and isolate the problem statement of this thesis. Even though the problem statement is general, environmental Web mapping applications were specifically selected for validating the research due to the societal relevance of the environmental issues. The perspective of providing a modest contribution to the society in the important issue of environmental management played also an important role in the personal motivation.

1.2 Problem Statement

There is an increased necessity for cartographic concepts and technologies suitable for environmental management. This can be attributed to the fact that the modeling and simulation of environmental processes often produces large spatiotemporal georeferenced data sets. However, the information analysis and interpretation efficiency is not advancing at the same rate as data acquisition thus creating a “data crisis” [Van Dam et al., 2000], and this is a bottleneck in understanding the data. In this context, graphical representation of numerical data is a key element for the comprehension of complex environmental processes. The graphical representation of the data in a map is often more adequate and helps gaining better insights and detecting spatial patterns such as, for example, pollution distribution and its impact on population, in comparison with looking at raw or tabular data. Maps and cartographic visualizations are powerful techniques for knowledge discovery, especially because they address the highly developed human skills of pattern recognition. Therefore analysis, planning, forecast and ultimately decision making in the environmental management domain are facilitated by displaying the environmental information in map form and in real-time.

As aforementioned, in the field of risk management many decisions must be taken on short notice based on cartographic visualizations. In order to provide the latest view of the problem the most up-to-date data from various systems has to be collated. Maps are therefore required to immediately reflect the updates in the data, and this without sacrificing cartographic quality. In this context, the map production directly from geodata sources opens new challenges for cartography. In case of a disaster, where life and property are in stake, cartographic visualizations must occur **automatically** from the available data sources. Moreover, also the distribution of the generated maps must be **performed on demand over the Web via services**, in order to provide an almost instant access to the map content independent of the physical location of the users. Consequently, a cartographic service that is able to create high quality maps based on open and reusable map descriptions from current GIS data is desirable. The following related research projects are used to demonstrate the requirements for such a service.

1.3 Related Projects

The problem of automatic generation of maps in real-time over Web services was also the high-level cartographic requirement of the European Integrated Project ORCHESTRA (Open Architecture and Spatial Data Infrastructure for Risk Management). The aim of ORCHESTRA was to improve the efficiency in dealing with risks by developing an open service architecture for risk management [Klopfer et al., 2008; ORCHESTRA Website, 2010], and therefore to provide an information infrastructure which is capable of managing risk information across Europe. By solving the problem of coupling different systems so they could interoperate, ORCHESTRA fostered the emergence of networks of risk management systems and endorsed the use of map-based Decision Support Systems (DSS) over the Web. These systems are providing the most recent information to the decision maker in map form and were envisioned to support the decision process in many phases of the disaster management cycle shown in *Figure 1*.

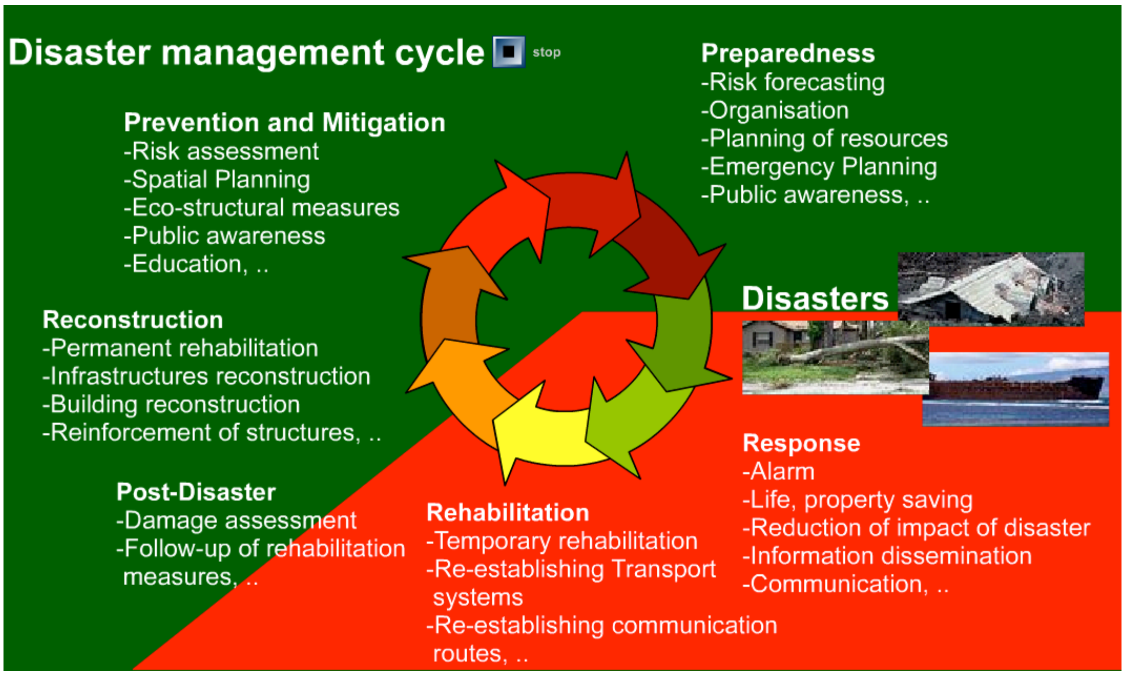


Figure 1. The Disaster Management Cycle as considered in the ORCHESTRA project [ORCHESTRA Consortium, 2007]

Among these phases, ORCHESTRA focused on the prevention and preparedness phases, with high interest in building pilot applications for risk assessment and forecasting. An example is the Pan-EUropean assessment of Natural HAzards (PEUNHA) for forest fires [Klopfer et al., 2008]. The PEUNHA pilot application demonstrated the creation of forest fire hazard maps, forest fire damage maps, and – through their combination – forest fire risk maps in order to support decisions on measures that could be taken for risk prevention on a European scale. The PEUNHA pilot required direct cartographic visualization of computed datasets (e.g. the likelihood of forest fires), and these datasets were generated by on-demand by geoprocessing services.

Additional requirements that strengthened the research subject of this thesis were provided in the frame of the European Integrated Project Sensors Anywhere (SANY). Accurate scientific conclusions, forecasting or validating environmental models require widespread temporal and spatial monitoring as well as an appropriate visualization of the acquired information. The SANY project was based on the results of the ORCHESTRA project and aimed to create a service architecture capable of providing a fast and cost-efficient way to reuse data and services from currently incompatible sensor and data sources [Klopfer et al., 2009]. In addition, one of the major objectives of the project was to develop advanced data fusion and decision support making use of the new architecture and thus providing added value to end users. Similar to other sources of environmental data, sensor data and also the results of scientific analysis and fusion algorithms must be dynamically presented to the decision makers in an appropriate form that boosts their interpretation. Sensor data serve as an extreme example for the dynamic nature of the data that must be visualized in real-time. According to these considerations, a service for the visualization of sensor data on maps, together with additional topographic and thematic data, was considered a priority also in the SANY project. With their requirements for the generation of maps directly from data via Web services, the ORCHESTRA and SANY projects helped to define the research questions of this thesis. Finally, it must be also mentioned that these requirements are also recurring in newer environmental projects. Such projects, discussed in chapter 7, reinforced the initial problem statement and have profited of the developments described in this thesis.

1.4 Research Questions and Objectives

This research started from the overall hypothesis that **automatic generation of maps from known GIS data is possible**. However the automatic generation of maps is in direct contradiction with the traditional cartographic workflow, which is highly dependent on manual or semiautomatic data selection and symbolization.

The first premise that shaped the research questions of this thesis was that a certain part of the data to be represented on the map might be changed or updated on a continuous basis. Consequently, there is no possibility to prepare and symbolize the data manually in order to produce the map in an acceptable timeframe. The map generation must therefore occur automatically. This requirement raises the first research question, namely if it is **possible to formally describe the map symbolization** in such detail that will allow the map to be created directly from the data without manual intervention.

As already mentioned, in environmental management there is a multitude of data sources that can be visualized on maps and this serves a variety of purposes, including risk assessment and decision support. Even when considering only the risk assessment, the map content can highly vary according to the depicted risks and the intended use. This premise raises the second research question, namely if a **map description** can be **generic and expressive** enough so that it can describe maps for a variety of purposes and intended uses while offering the same symbolization possibilities as available in a graphics software.

Finally, considering that in some use cases for environmental monitoring and crisis management the map must be generated and made available in real-time, the third research question arises, whether it is possible to create a **Web-based service-oriented** mechanism to produce maps that can be distributed on demand over the Web.

The above research questions played an important role in formulating the overall objective of this thesis: developing a way of creating on demand cartographic visualizations directly from source data in a generic service-oriented manner, concisely expressed in terms of concepts and technologies for **cartographic Web services**.

1.5 Thesis Structure

The thesis is organized as follows. In the first part of the thesis, chapter 1 introduces the problem statement and the research objectives, chapter 2 presents the background information and the state of the art in Web cartography and chapter 3 makes an analysis of the related work that greatly influenced the present research.

This introductory part is followed by the core of the thesis represented by chapter 4, which details the map representation language for cartography and allows the syntactic formalization of the map content, and chapter 5, which focuses on the specification of cartographic Web services. Furthermore, chapter 6 introduces the implementation of the cartographic Web services in prototype software and chapter 7 proves the core thesis concepts by presenting several real-world cartographic applications designed with a service-driven cartographic workflow. Finally, chapter 8 mentions the additional research performed in relation with this thesis and chapter 9 presents the synopsis of results, the main conclusions and the outlook of this thesis.

2 State-of-the-Art

This chapter sets the focus of this research on the Web, on service-oriented architectures and on reusing standards in order to answer the fundamental questions of this thesis. A review of the relevant technological progress in Web Cartography and GIS is followed by an analysis of existing map representations. The latter indicates the direction of the work to be performed in order to fulfill the goal of this thesis and to answer the research questions presented in the first chapter. Finally, it motivates our choice to rely on standards and to enhance these with the latest developments in Web Cartography and GIS.

2.1 The Internet and the Web

The Web is the medium that allows an almost instant access to a map, independent of the physical location of the users and the available hardware. Because we focus our research on the Web environment, we assure that the distribution of the generated maps can be done automatically over a Web browser.

In order to support the definition of cartographic Web services, we will start by providing in the following lines a brief overview of the Web environment and its underlying infrastructure.

2.1.1 The Internet

To begin with, a computer network is defined as a collection of autonomous and possibly heterogeneous computers interconnected by a single technology [Tannenbaum, 2002]. We say that two computers are interconnected if they are able to exchange data. There exist many networks in the world, often with different hardware and software. Accordingly, the global collection of interconnected networks is called an internetwork or the Internet.

2.1.2 The Web

In order for the Internet to be useful, it is required to have also a distributed system for exchanging information [Tannenbaum, 2002]. A distributed system is a way of presenting a computer network to its users as a single coherent system. The success of the Internet was due to exactly such distributed systems, as for example mail, news, remote login, file transfer, and the World Wide Web (WWW). Initially developed for research purposes, the WWW (or shortly the Web) changed its character to a public utility, a fast and cheap communication channel for information exchange, available everywhere. Also in the case of cartographic products the distribution barriers have fallen, since the Web offers ubiquitous, direct, convenient and multi-user parallel access to cartographic products from all over the world. Therefore, the Web is currently the most auspicious medium for exchanging and distributing cartographic information.

2.1.3 The Internet vs. the Web

To summarize, the Internet is a network of networks and the technological enabler of the Web. The Web is a distributed system that runs on top of the Internet and, with respect to the subject matter hereof, it is currently the most appropriate communication channel for on-demand map distribution. These previously named concepts are fundamental for understanding Web services.

2.2 Service Oriented Architectures and Web Services

The Web defines a mechanism for a site to set up a number of pages of information with embedded links to resources such as other pages, text, images, videos and various other files. All the links are clearly defined by Uniform Resource Locators (URLs). The resources identified by URLs can be static or dynamic. Examples of static Web resources are HyperText Markup Language (HTML) files available on Web servers. But static content get outdated relatively quick. Dynamic Web resources are generated on-demand every time their URL is accessed. This implies more than just HTML files. It requires a software infrastructure providing services that automatically generate the content based on the latest information available. Dynamic content is present in our everyday life, such as news aggregators, weather forecasts, flight booking and online shopping. The concept of dynamic content is central to this research, considering that the goal of this thesis is to automatically generate maps from spatial data that might be changed or updated on a continuous basis. For the end user however, there is no perceived difference between accessing a static document and accessing a service on the Web.

2.2.1 Service Oriented Architectures (SOA)

Service Oriented Architectures (SOA) and Web services have an important role in generating dynamic Web resources. Conceptually, SOA is an architectural style based on loosely coupled interacting software components that provide services. Furthermore, services are the central elements in developing any kind of software applications based on SOA. Consequently, the SOA applications are composed of interconnected services available over a distributed network. They allow for seamless information integration by abstracting the complexity of the heterogeneous nature of the Internet and the Web.

2.2.2 Web Services

A service can be defined as a piece of functionality made available by a service provider in order to deliver end results for a service consumer [Alonso et al. 2004]. They build upon the basic concept underlying the Web, the concept of request-response presented in *Figure 2*. A service consumer (client) sends a service request to a service provider (server). The service provider returns a response to the service consumer containing the desired results.

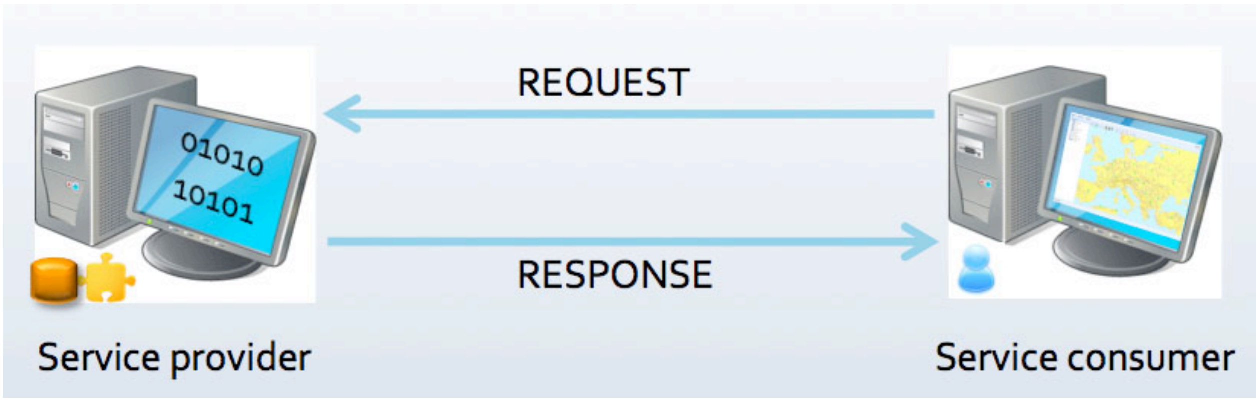


Figure 2. Request-response mechanism

SOA can be applied in various ways, but usually SOA is implemented with Web services. A Web service is a type of service that can be identified unambiguously by an URL and uses Web standards such as HTTP for transport.

2.2.3 Interfaces and Messages

SOA enforces the concepts of **interfaces and messages**. An interface constitutes a contract defining the functionality of the service in a platform-independent manner. Interfaces are defined for all participating services and they should be universally available for all providers and consumers.

Services communicate with consumers by using messages. Messages are described and delivered through the interfaces. The service interface defines the types of messages a service can process. To achieve platform and language independency, messages are typically constructed using XML documents that comply with the corresponding XML schemas. The purpose of the schemas is to limit the vocabulary and the structure of messages. Moreover, XML schemas are extensible which allows new versions of services to be introduced without modifying the existing ones.

2.2.4 Characteristics of Web Services

Services exhibit several other properties that must be mentioned for a proper understanding of the cartographic Web services. The interaction is loosely coupled, meaning that the services do not share common modules or data model with a service consumer. By default, Web services are stateless, meaning that each interaction can be performed independently, without knowing the current conditions of the service (note: there are additional frameworks or technologies to enable state transfer though). Finally, the usage of services is location transparent, e.g. clients do not have to know if the service is local or only accessible over a network. These properties enable and support efficient composition of cartographic Web applications based on Web services.

To summarize, Web services work according to a request-response mechanism, are defined in terms of interfaces and messages, and are characterized by loosely coupled, stateless and location transparent interactions as shown in *Figure 3*.

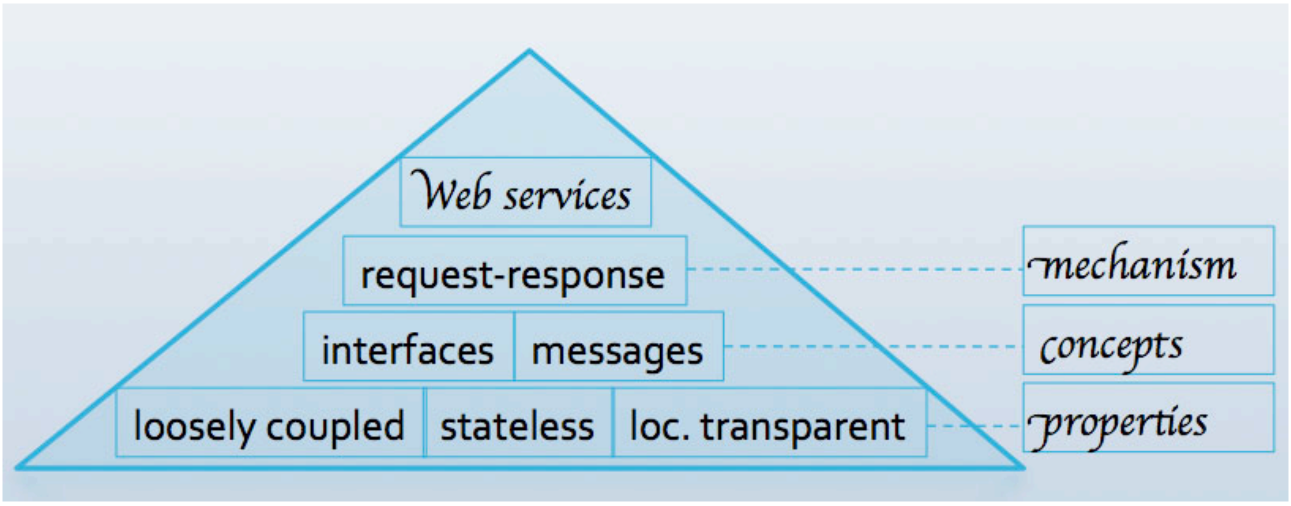


Figure 3. An overview of Web services

2.2.5 Web Services with SOAP and WSDL

Regarding the technical implementation of Web services, there are currently two major implementation styles: “big” Web services and REST (REpresentational State Transfer) [Richardson el al., 2007]. The World Wide Web Consortium (W3C) technically defined Web services, quote: “A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.” [W3C Web Services Architecture Working Group, 2004]. Web Services Description Language (WSDL) is an XML language for unequivocally defining the interface of a service. Simple Object Access Protocol (SOAP) is an XML language defining the message format with a top-level XML element called “envelope”, which contains the content of the message in a “body” child element. The combination SOAP/WSDL has gained strong industry support and is a standard way for doing Web services.

2.2.6 REST Web Services

Recently, the REST Web services have gained widespread acceptance across the Web as a presumed more simple alternative to SOAP/WSDL “big” Web services [Pautasso et al., 2008]. It was originally introduced for building large-scale distributed hypermedia systems and it is normally used in conjunction with HTTP. It has four technology principles, two of them being the most important. The first principle is the resource identification through an Uniform Resource Identificator (URI), which effectively enforces a global addressing space. The second principle is the uniform interface definition through the use of the well-established HTTP verbs: GET retrieves the current resource, PUT creates a new resource, POST is used to update a resource and DELETE can be used to delete a resource.

2.2.7 SOAP/WSDL vs. REST

There are many ongoing discussions about the strengths and weaknesses of SOAP/WSDL vs. REST Web services [Pautasso et al., 2008]. However, the presented principles for Web services are generic. Therefore, for defining the cartographic Web services in chapter 4, we have decided to present the service interface in an abstract manner, independently from any specific implementation styles. These abstract interfaces can then be implemented either in SOAP or REST Web services style.

2.3 Related Work in Web Cartography and Web GIS

[Cartwright, 2007] and [Peterson, 2007] give a thorough review of the area of multimedia and Web cartography and of the theory supporting Internet cartography in general. In the following we will restrict ourselves to an overview of the most recent related work in the field of Web cartography.

2.3.1 Digital Map Production

The availability of inexpensive personal computers revolutionized the management of geographic information and the map production. Geographic information systems (GIS) emerged as the software solution for the integration of geographic information within computerized information systems and cartographers gradually migrated to computer aided design (CAD) systems, GIS, and desktop publishing (DTP) systems for digital map production. These computerized systems were the catalysts of both digital and interactive cartography [Hurni, 2008].

2.3.2 Electronic Atlases

Much interest revolved around the production of electronic atlases since the late 1980s with the availability of Apple's HyperCard software [Cartwright, 2007], since the technological advancements that caused the transition from analog to digital cartography also triggered the development of interactive electronic atlases. Even though early electronic atlases had limited functionality, such as zoom, layer selection and name search [Hurni, 2008], they evolved from closed monolithic applications to open distributed systems in which real-time map rendering components and services play an important role [Sykora et al., 2007].

2.3.3 Atlas Information Systems

Electronic atlases grew in thematic complexity and interactive functionalities. Atlas information systems (AIS) are state of the art cartographic applications responsible for visualizing and analyzing organized thematic collections of spatial data. Their distinctive features lie in their high cartographic quality, their user-friendliness and their effectiveness as communication platforms since they present geographical information in an interactive, multi-scale manner unachievable by printed Atlases.

Recent AIS research promotes a decoupling between the map and the Graphical User Interface (GUI). The purpose of this is to create more flexible, database-driven architectures for multimedia cartographic applications in general [Gogu et al., 2005; Sykora et al., 2007]. Example implementations are provided by GEOWARN – Geospatial Warning Systems – project with its AIS for volcanic monitoring [Hurni et al., 2004; Gogu et al., 2005] and STATLAS – Statistical Atlas Project of the European Union – with a distributed AIS providing access to visualizations of Europe-wide statistical data [Sykora et al., 2007].

2.3.4 Multimedia Atlas Information Systems

The crowning of electronic atlas evolution is represented by Multimedia atlas information systems (MAIS). They have become information platforms, which allow a user-oriented communication for information and decision-making purposes through systematic, targeted collections of spatially related knowledge in electronic form [Hurni, 2008]. MAIS integrate visualization in 2D and 3D mode, and have special interactive functions for geographic and thematic navigation, querying and analysis. Moreover, related multimedia information such as text, images, videos, audio documents, animations, graphics and diagrams are linked to the geographic entities in order to provide a truly informative and immersive user experience.

2.3.5 Towards Web Cartography

Already mid 1990's, the evolution and the widespread use of World Wide Web (WWW) initiates the field of Web cartography. The developments in this area started, as mentioned, immediately after the emergence of the Web. In the beginning, this new medium was used solely for fast map distribution. Tourist and public transportation maps are among the first types of maps that could be downloaded from the Web and then printed at home [Richmond et al., 2003; Mooney at al., 2003].

The cartographic community immediately recognized the potential of the Web and pleaded for high quality interactive Web mapping with the creation of the ICA (International Cartographic Association) Commission on Maps and the Internet [ICA, 2010].

2.3.6 Interactive Web Cartography

Web mapping based on the eXtensible Markup Language (XML) [Letho, 2003; Zazlavsky, 2003] and Scalable Vector Graphics (SVG) [Neumann 2005; Iosifescu et al., 2006] technologies are the most recent examples of cartographic developments, which take advantage of the latest Web technologies.

XML is a World Wide Web Consortium (W3C) standard defining a simple and very flexible text format for the exchange of a wide variety of data on the Web [W3C, 2010]. SVG technology is an XML-based W3C standard that integrates vector and raster graphics, multimedia, text, animations and scripting. The vector graphics presentation combined with dynamic content manipulation is particularly interesting for cartographic purposes and interactive Web mapping. In addition, SVG introduces the painter's model of rendering [W3C, 2003]. In this context, paint represents a way of putting color values onto the canvas and it consist of both color values and associated alpha values which control the blending of colors against already existing color values on the canvas. For cartography it is important to notice that paint is applied in successive operations and that each operation paints over some area of the canvas. When the new area overlaps a previously painted area the new paint partially or completely obscures the old.

The ECMAScript language (also known as JavaScript) enables the manipulation of the document content thus allowing the development of interactive applications. Elements in the SVG applications are receptive to mouse and keyboard interactions, which may trigger one or more

JavaScript functions [Neumann, 2005; Iosifescu et al., 2006]. These cartographic developments, albeit allowing for highly interactive Web mapping applications, are available as closely coupled, all-in-one frameworks.

2.3.7 Web Cartography and GIS

The trend for the integration with the Web is common to all geographic and atlas information systems. In this respect, the previous efforts regarding the integration of geographic information with the Web through Web maps, Atlas Information Systems (AIS), and Geographic Information Systems (GIS) provide the foundations for current research. The geographic information science and GIS contribute to the realm of Web Cartography with the technical standardization of geographic information and with the potential of querying maps and geographic information through the Web. Moreover, notable existing national atlases are following this trend and have seized the opportunity to evolve towards Web-based architectures (e.g. Atlas of Canada, the Swiss World Atlas Interactive) or are planning on creating an online platform (e.g. Atlas of Switzerland) [Sieber et al., 2011].

2.3.8 Online GIS Services

Finally, from the point of view of Web Cartography, a great promise comes from the development and subsequent standardization of GIS services. In an attempt to disseminate elements of desktop GIS to a larger audience over the Web, all major GIS vendors created their own Web mapping systems. However, most of these interactive mapping systems were developed as a set of proprietary implementations. To address the interoperability problems of existing online mapping services, the Open Geospatial Consortium (OGC) undertook extensive work on standardization, aiming towards open standards and giving transparent access to heterogeneous geo-data and geo-processing resources in a networked environment, including a non-proprietary Web mapping approach based on open interfaces, encodings and schemes. The standardization effort was worthwhile, which is proven by the wide industry acceptance of the Web Map Service (WMS) [OGC, 2000; ISO, 2005] and Styled Layer Descriptor (SLD) standards [OGC, 2002c, 2005a]. The success can be demonstrated by the implementation of these standards in current well-known GIS software, such as the ESRI ArcGIS software family. These standards define a service interface to enable the **visual overlay of distributed geographic information maps** simultaneously, over the Internet. The potential of WMS for online cartography was recently perceived and analyzed by the cartographic community [Cammack, 2007; Sykora et al., 2007].

These developments, although of separate origins, demonstrate that the trend for the integration with the Web is common to all geographic and cartographic information systems.

2.4 Overview of Relevant Geospatial Standards

2.4.1 Standards

This chapter concludes with an overview of relevant geospatial standards. Standards establish *technical specifications* to achieve the required level of compatibility, interchangeability, or commonality to obtain interoperability for a specific domain or purpose. Standards are developed and maintained by well-established standards organizations or standards bodies such as for instance the International Organization for Standardization (ISO), the World Wide Web Consortium (W3C), the Institute of Electrical and Electronics Engineers (IEEE) or the Open Geospatial Consortium (OGC).

2.4.2 Standards Organizations

The W3C is the main standards organization for the World Wide Web (WWW) developing Web standards and guidelines. W3C, as an industry consortium, ensures compatibility and agreement when adopting new standards. Organizations from the whole world can join the W3C in vendor-neutral forums to create Web standards. Starting inn 1994, the W3C has published more than 90 such standards, called W3C Recommendations, for instance HTML or XML.

Regarding standards in the geospatial domain, the main contributors are currently the Technical Committee 211 (ISO/TC 211) formed within the International Organization for Standardization (ISO) and the Open Geospatial Consortium (OGC). The ISO/TC 211 is a standard technical committee covering the area of geographic information and geomatics. It is in charge of the 19100 series of standards mainly defining the structure of the exchanged data. The Open Geospatial Consortium (OGC) is dedicated to develop standards for geospatial content and services. The OGC Specification Program and Interoperability Program provide an industry consensus process to plan, develop, review and officially adopt OGC specifications for interfaces, encodings and schemas that enable interoperable geoprocessing services, data, and applications.

These organizations are following different, but complementary approaches. While ISO/TC 211 is defining mainly the structure of the exchanged data, OGC defines a framework specifying standardized interfaces supporting the development of interoperable software components for the access and processing of spatial data. ISO/TC 211 applies a top-down approach while OGC uses a bottom-up perspective. To avoid competing and inconsistent standards the organizations collaborate closely with each other [Gronmo et al., 2000; Wytzisk, 2003].

2.4.3 Simple Features Specification

The first OGC Specification to be approved by the membership was the Simple Features Specification, released in 1997. This standard is the foundation for the interoperability and exchange of GIS data. It specifies the interface that enables GIS-enabled systems to interoperate in terms of "simple features." The supported geometry types include points, lines, curves, and polygons, all based on two-dimensional geometry. Each geometric object is associated with a Spatial Reference System, which describes the coordinate space in which the geometric object is defined. The standard is implemented in virtually all GIS programs and spatial extensions for databases, both commercial and open source. This highlights the fact that simple geometries represented the first milestone required for the interoperability in the geospatial realm to succeed.

2.4.4 Web Map Server (WMS)

In 2000 the Web Map Server Specification was released in its 1.0 version [OGC 2000]. The Web Map Server (WMS) Specification standardizes the way in which Web clients request maps. It defines three operations: GetCapabilities (returns service metadata), GetMap (returns a map as digital file) and GetFeatureInfo (optional operation that returns feature information at a specified location). The standard WMS operations can be invoked using a WMS client by submitting requests in the form of HTTP Uniform Resource Locators (URLs). Each WMS operation has several mandatory or optional request parameters with standardized names. In the case of a GetMap request, the URL contains information about the specific attributes of the invoked request (e.g. version and name), the output image characteristics (e.g. format, width and height), what information is to be shown on the map (layers), how this information is represented on the map (styles), what geographical region is to be mapped (bounding box) and the desired coordinate reference system (CRS). The main output of the request is a map, which can be either in raster (e.g. jpeg, png) or symbolized-vector format (e.g. SVG). A service does not provide a graphical user interface, however, depending on the implementation, a geographic viewer, another map service instance, or even a Web browser could act as a WMS client. The WMS, in its 1.3 version, became also an ISO international standard for Web map services in 2005 [ISO, 2005].

2.4.5 Relationship between WMS, Styled Layer Descriptor (SLD) and Feature Encoding Specification (FES)

In 2002, four important specifications were released: the Styled Layer Descriptor (SLD) [OGC, 2002c], the Filter Encoding Standard (FES) [OGC, 2002a], the Web Feature Service (WFS) standard [OGC, 2002d] and the Geography Markup Language (GML) international standard [OGC, 2002b].

The SLD standardizes how the WMS specification can be extended to allow user-defined symbolization of feature and coverage data. SLD is an important standard from a cartographic perspective, since it encourages the dynamic creation of Web maps directly from geographical features while offering basic mechanisms for customizing the rendering of the map features. FES enhances the functionality of WFS, WMS and SLD with an XML encoding for filter expressions that logically combines constraints on the properties of a feature in order to identify a particular subset of features to be operated upon. An example usage in the context of the WMS/SLD is to provide constraints specified on values of spatial, temporal and scalar properties, thus enabling the service to identify a subset of features and render them in a particular style.

2.4.6 Web Feature Service (WFS) and Geography Markup Language (GML)

The GML Encoding Specification is an XML encoding for the modeling, transport and storage of simple features, including the spatial and non-spatial properties of geographic features. The WFS specification defines interfaces for data access and manipulation operations on geographic features, using HTTP as the communication protocol. The WFS and GML are closely interconnected: WFS allows a client to retrieve and update geospatial data encoded in GML. Via these interfaces, a Web user or service can combine, use and manage GML-encoded geodata from different sources. Moreover, WFS/GML may be used to serve the features composing a map image in a WMS.

2.4.7 Standards Evolution, from SLD to Symbology Encoding (SE)

Since their original publication, all the above-mentioned standards evolved in new versions [OGC Standards Website, 2010]. Finally, in 2007 the Symbology Encoding (SE) standard was released, which is the direct continuation of the SLD in conjunction with the Styled Layer Descriptor Profile for the Web Map Service Implementation Specification [OGC, 2007a; 2007b]. SE is the most recent OGC standard for portrayal of geographic information. The SLD specification document was split up into two documents to allow the parts that are not specific to WMS to be reused by other service specifications under the umbrella of the SE standard.

2.4.8 The Value of Standards in the Development of Cartographic Web Services

The value of using standards can be explained in a more general manner with the advantages already brought by the existing standards related to geospatial data and Web mapping. They enabled a commonly agreed communication language for exchanging geospatial information, therefore reducing the chaos generated by incompatible offerings from multiple vendors. If a standards-based strategy is applied in the development of map representations, then it will increase the possibilities for the future exchange of maps and map representations between different software packages. Detailed information for choosing a standards-based approach for research and development (R&D) can be also found in a training unit of the ORCHESTRA Project [ORCHESTRA Consortium, 2007]. This reference can be used to enhance the understanding of existing standards beyond the information provided in this subchapter.

2.4.8 Standards as a Foundation for Cartographic Web Services

The presented standards are the foundation of the research in cartographic Web services. The WMS specification supports the ability to access a preset collection of map layers, thus providing an accepted interface specification for map access over the Web. But in WMS the users have no possibility of defining their own symbology. The SLD and SE fill this gap and offer a basic styling language that allows a client to customize how a layer will look like. Finally, the FES, GML and the Simple Features provide a generic representation of GIS data that is independent of a specific model or format. As these standards were designed for GIS data exchange and not for map representation, they leave significant freedom for cartographic research. Moreover, they provide the overall framework for bridging GIS and cartography.

3
Towards a Map Representation

This chapter describes the map-making process and the need for a map representation as prerequisite for cartographic web services. The focus lies on the review and analysis of existing map representations. Previous work is analyzed in order to build a foundation for defining a map representation for Cartography.

3.1 The Map Representation

When considering the map-making process, we might think of an “Universal Cartography Manual” that would contain strict instructions for making perfect maps in all situations. Such a hypothetical manual would allow anyone to create cartographically correct and expressive maps just by following fixed, simple and clear steps. However, the more one dives in the study of cartography as a science, the more it becomes clear why such a manual does not exist. Cartography is not only science, but also creativity and art. Moreover, there is no unique solution for creating a map product. Given a data set, there are many ways to visualize the data with cartographic correctness. Not only that the creation of a new map product may pose unique challenges that a cartographer must analyze and solve, but there is more than one approach in solving them, depending on the art and the innovation of the cartographer.

3.1.1 Map Representation as a Description of the Map Making Process

One way of learning cartography is learning-by-doing: good versus less-than-optimal cartographic examples are compared and analyzed. Lacking an “universal cartography manual” as mentioned above, this work analyzed the complexity of maps and their inherent exceptions by looking at various map examples and trying to understand how problems where solved. One example was the printed Topographic Map of Methana. The work, performed by [Hurni, 1995], introduced, for the first time at the Institute of Cartography, a complete computer-assisted cartographic workflow. The Topographic Map of Methana was produced solely by digital methods, and the map compilation process is described in-depth [Hurni, 1995]. The documentation of the map compilation process allows every cartographer to follow the various cartographic choices and to understand that in every cartographic product there are as many exceptions as there are cartographic principles. The exceptions are caused by the unavoidable complexity of combining several data layers and ultimately it is the “art” of an experienced cartographer that applies the cartographic principles every time in original ways and insures that the map message is easily conveyed to the map user.

However, very few printed map products are documented in such detail. Therefore, it would be inherently useful for understanding the cartographic practice if cartographic choices would be described in the corresponding map representation for every digital map, thus effectively documenting the map-making process.

3.1.2 Definition of the Map Representation

We define the notion of a map representation as the description of a map, in a specific form, and with sufficient amount of detail in order that a software is able to generate the corresponding map rendering from the geospatial data. More specifically, in this thesis we consider that the map representation formally describes all the steps necessary for creating the map content and enables a cartographic web service to render a corresponding map and display it to a user.

From the cartographic point of view, the map representation mainly defines the symbology that is to be applied to every feature from the dataset that is to be visualized in the map. It is possible to express a map representation because it can be defined with a set of statements that are independent of the physical model of the data. Consequently, the specific format in which the spatial data is available is irrelevant and only the logical model of the data is used to specify the representation.

3.2 Existing Map Description Languages

The core research question of this thesis is to formally describe the map in such detail that it will be possible to generate the map automatically from GIS data, based solely on the map representation (the data is assumed to be primed for standardized scales and degrees of generalization). As consequence, existing map description languages defining the desired map symbolization adapted to the area of distributed mapping represent **important cornerstones towards a formal and generic map representation**.

3.2.1 Selected Map Description Languages

The analysis of the relevant state of the art on existing map representations was the starting point of the research performed in this thesis. Among existing work, three relevant map description languages for the topographic and thematic representations were selected for analysis: the Cartographic Markup Language (CartoML), the Diagram Markup Language (DiaML) and the Symbology Encoding (SE). These were selected based on their characteristics of being open, human readable, expressive and informative. The openness of the map representation was one the most important criteria as it permits users to study, change and improve upon it without the hurdles of an undocumented proprietary format. This was the main reason why proprietary map descriptions such as the one of the new cartographic tools in ArcGIS were not suitable for in depth analysis. Human readability insures its direct interpretation and editing without the help of additional computer software. It means that even without prior knowledge of the format, one can read the content and has a reasonable chance of understanding it. Cartographic expressivity was also an important selection criterion; the analyzed map representations must cover a wide spectrum of different cartographic symbolizations. Finally, a map representation had to be informative; meaning that the construction of the map can be understood from the map representation and that there are minimal hard-coded behaviors incorporated in the software that uses the corresponding map representation.

3.2.2 CartoML

The CartoML is a map description language designed by cartographers for the use in the iMap visualization environment of the already mentioned STATLAS project [Sykora et al., 2007]. Although STATLAS was not designed for the Web, CartoML illustrates the state of the art efforts of cartographers in the area of open distributed processing. It is an approach for managing various thematic map types by packaging cartographic information into an easily readable and understandable XML-based document. The visualization environment (iMap) of STATLAS and CartoML were designed to go beyond the basic drawing of thematic maps. Some of the advanced features include object transparencies, anti-aliasing, Bézier curves, advanced point symbols and thematic representations as illustrated in *Figure 4*. Moreover CartoML uses JavaScript, a scripting language for the description of symbols and diagrams. With the scripting language one can define for example new diagrams based on fundamental graphic primitives.

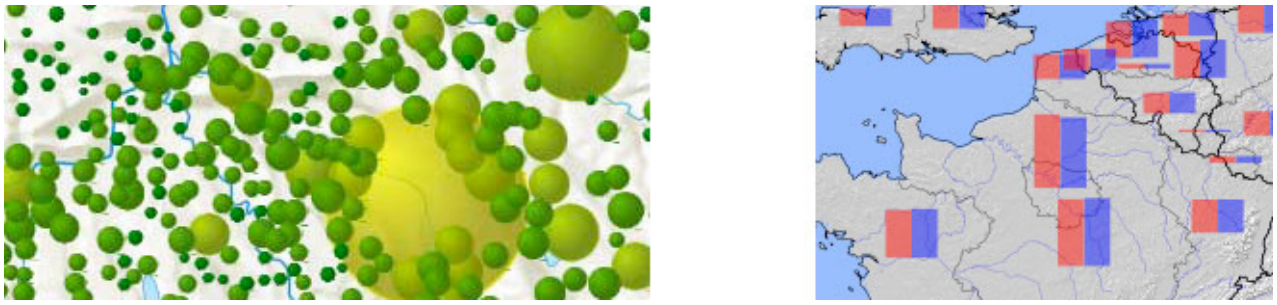


Figure 4. Representations created with CartoML and iMap [Sykora et al., 2007]

CartoML syntax is exemplified with a basic symbolization of line features, given the attributes width and color (including opacity), as presented below:

```
<symbolization>
        <strokeattribute>
           <linewidth>0.05 mm</linewidth>
           <color>#ff000000</color>
        </strokeattribute>
        <table name="pop">
           <var>change</var>
           <manual_classification>-100 -50 -20 0 20 50 100</manual_classification>
           <colortable xlink:href="../../colortables.xml#blue_red_4:4"/>
           <fillattribute>
              <color>#55ffffff</color>
           </fillattribute>
        </table>
</symbolization>
```

3.2.3 DiaML

DiaML is another XML-based description language specific for complex thematic point symbols [Schnabel, 2007]. While current cartographic interfaces provide a finite subset of the large number of potential symbol definitions, DiaML uses a construction model to generate as many different symbols and diagram types as needed for the cartographic description. This model is based on an analysis of the structure of cartographic symbols for the portrayal of statistical data. In addition, the MapSymbolBrewer software was developed to interactively design map symbols and to export the symbols as DiaML descriptions. The MapSymbolBrewer can also present symbols of statistical data on a map as shown in *Figure 5*.

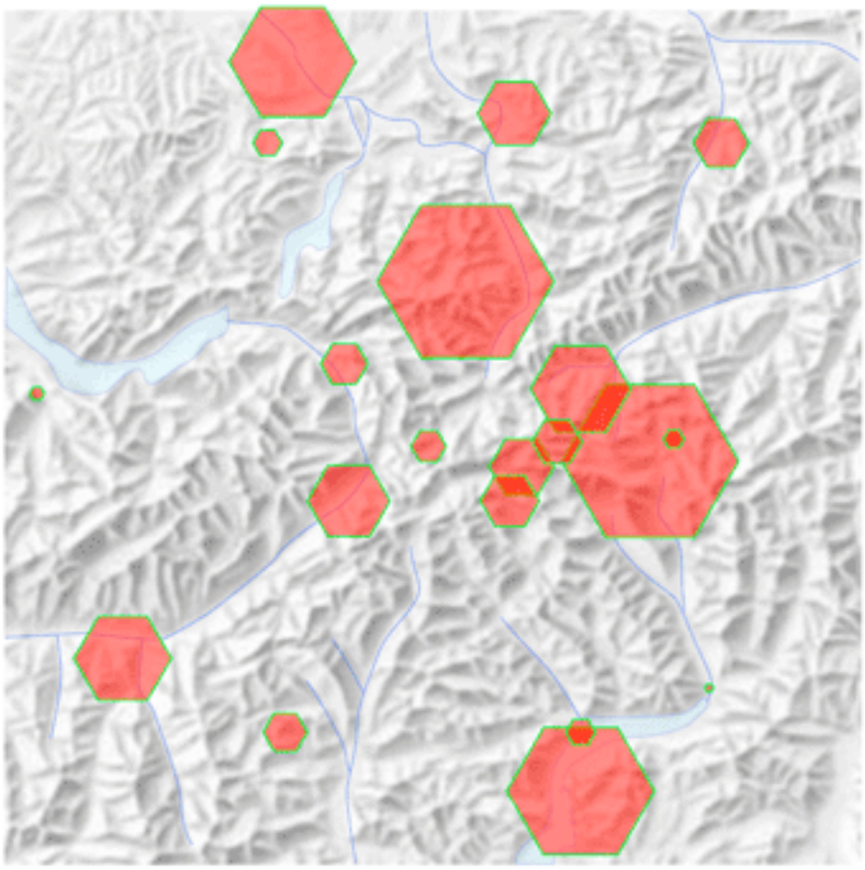


Figure 5. Representation created with DiaML and MapSymbolBrewer [Schnabel, 2004]

In DiaML ten cartographic primitives can be defined: point, polyline, curve, ellipse, circle, sector, ring, ring sector, rectangle (bar), and regular polygon. Each of the arranged cartographic primitives can be scaled in one or two directions to visualize the data values. Additional map symbol properties such as transformation parameters (e.g. rotations), guides and labels as well as graphical primitive properties (styles) are available. A DiaML syntax example is presented below:

```
<symbol>
   <!-- STYLE DEFINITIONS -->
   <style id="s1">
      <fill>red</fill>
      <stroke>grey</stroke>
   </style>
   <style id="s2">
      <fill>orange</fill>
      <stroke>grey</stroke>
   </style>
   <!-- PRIMITIVE DEFINITIONS -->
   <primitive id="p1">
      <sector proportional="area">
         <r scale="totalSum">1</r>
         <angle scale="dataValue">1</angle>
      </sector>
   </primitive>
   <!-- SYMBOL DESCRIPTION -->
   <diagram minSize="5" areaRatio="10.0">
      <diagramArrangement><polar/></diagramArrangement>
      <diagramRelation>
         <dataRef>pop96</dataRef>
         <dataRef>pop97</dataRef>
         <dataRef>pop98</dataRef>
         <primitiveRef>p1</primitiveRef>
      <styleRef>s1</styleRef>
    </diagramRelation>
    <guides/>
    <labelData>yes</labelData>
  </diagram>
</symbol>
```

3.2.4 Symbology Encoding

The third relevant map description language originates from the Open Geospatial Consortium (OGC). The Symbology Encoding (SE) is an XML-based standardized description language for defining layer symbolizations in a generic manner [OGC, 2005b]. Although SE is independent of any service specifications, it is usually used as part of the Styled Layer Descriptor (SLD) standard for allowing a user to change the symbolization of maps provided by OGC Web Map Services (WMS). The main advantages of SE and SLD are the generic structuring of the style attributes for computers as well as for users, and its standard status. Since OGC Web Services, including the WMS, are based on layers, the SLD allows each layer to be symbolized with user-defined styles expressed in SE with the final goal of changing the default map symbolization for basic topographic data according to the cartographic expectations of the user.

An example of an SLD/SE style that associates the SE-defined stroke color blue to the layer "hydrology" is presented below [OGC, 2005b] and illustrated in *Figure 6*.

```
<StyledLayerDescriptor version="1.0.0">
  <UserLayer>
    <Name>Hydrology</Name>
    <UserStyle>
      <Name>Blue_stroke</Name>
      <FeatureTypeStyle>
        <Rule>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#0000ff</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </UserLayer>
</StyledLayerDescriptor>
```

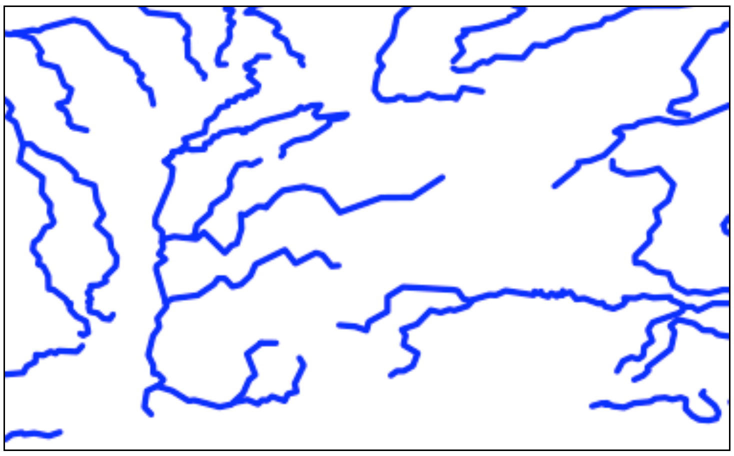


Figure 6. Representation of linear features created with SLD/SE and WMS [after OGC, 2005b]

The representation of geographical features with SLD is not restricted only to layers containing linear geometries (as shown in *Figure 7*), but it is also available for point, polygon and raster layers.

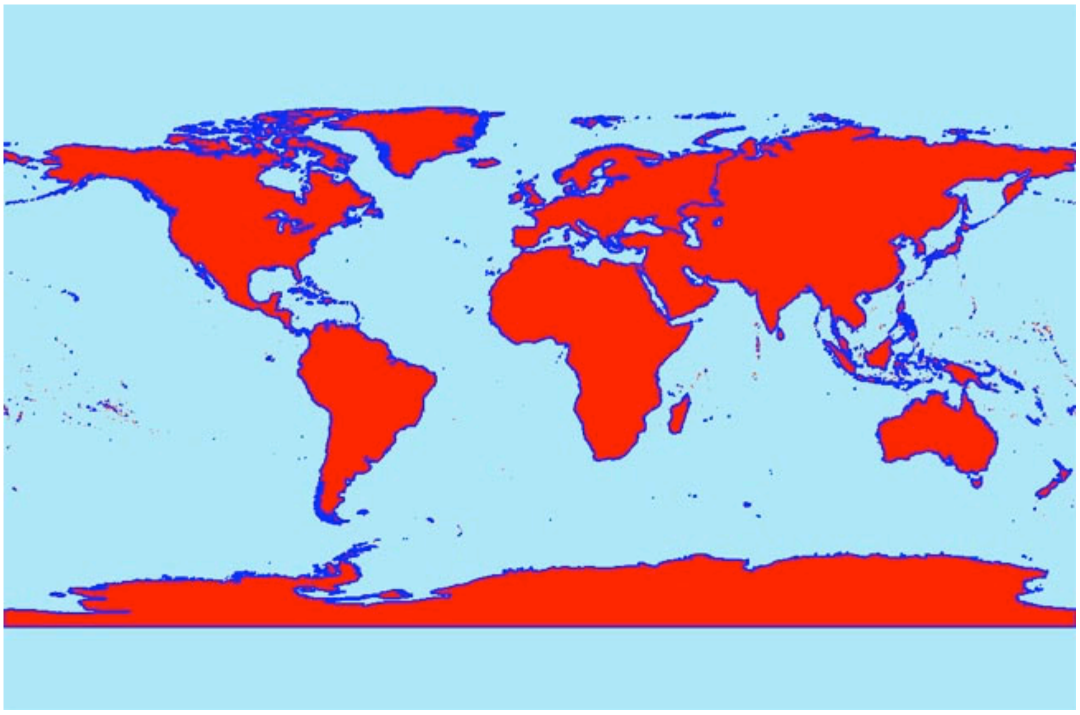


Figure 7. Representation of polygon features created with SLD/SE and WMS [e-cartouche, 2011]

Finally it must be noticed that CartoML, DiaML and SLD/SE are XML dialects. They also make use of a syntax inspired from the Cascading Style Sheets (CSS) for expressing symbols properties such as color, stoke or fill in a similar manner. It was a natural choice, considering that CSS describes a presentation semantics that can be applied to any kind of XML document.

3.2.5 Cartographic Representations in ArcGIS

ESRI recently improved the cartographic capabilities of their ArcGIS Desktop product. The cartographic tools of ArcGIS 9.2 introduced in 2007 illustrate very well the increased efforts of the industry to obtain an optimal representation of features available in geospatial databases. The main purpose of these new tools was to provide the flexibility to represent cartographic rules without creating additional datasets for cartographic purposes [Glennon and Glennon, 2007]. The cartographic tools also support a better implementation of cartographic symbolization in ArcGIS, with the expected effect of reducing or eliminating additional graphics software from the cartographic workflow for the production of printed maps. With this new approach, the symbolization is saved in the geodatabase alongside the geographic entities, and it can be freely changed without affecting the underlying geometry [Romer, 2010].

Along with the new tools, the concept of cartographic representations is introduced. Such representation rules can be created and edited graphically in ArcMap from scratch, or by converting symbolized layer (Symbology to Representation Conversion). Representations provide a new way of categorizing and displaying data based on rules that define the appearance of map features. They allow the editing of symbols without changing the actual shape or position of features in the database; therefore features are drawn based on the symbols and properties specified for a particular representation rule. Rules are fundamentally made of one or more symbol layers, i.e. strokes that draw lines or polygon outline, markers that draw points, or fills that draw polygons as a solid or with a pattern [Glennon and Glennon, 2007]. Exceptions to the representation rules (for specific features) are called overrides, and they can be defined for specific features within the editing mode from ArcMap. Furthermore, there is also the possibility for a "Field Override", where certain characteristics of the representation (e.g. colors or widths) can be parameterized with values taken from an existing attribute column of the geodatabase table [Romer, 2010].

Geometric effects can be also added to alter symbol appearance, e.g. dashing and offsetting a line or depicting a point feature as a polygon in order to construct a user defined point symbol. In addition, representations have an invisibility property that allows certain features to be hidden in case they should not be represented without deleting the actual features from the database. Moreover, for the ultimate freedom in changing the appearance of the representation, the user can use the Representations Toolbar from ArcMap and decouple any graphical object from the representation rules. These objects are then called "Free Representations" and can be changed arbitrarily, like in a graphics software [Romer, 2010].

With representations, cartographers have now the possibility to improve the map clarity directly in ArcMap. Moreover it simplifies data management as they eliminate the need to create and maintain redundant copies of data for cartographic purposes. Both representations rules and overrides are stored in a proprietary binary format and saved in attribute columns from the geodatabase table containing the geographic entities.

3.3 Analysis of Map Representations

3.3.1 Analysis of Map Representations Regarding Cartographic Expressivity

It can be observed that map representations designed by cartographers such as CartoML or DiaML are characterized by the expressivity and intuitiveness of the cartographic visualizations. For example CartoML uses Bézier curves, advanced point symbols and includes extensive support for thematic mapping. DiaML also allows modeling of complex point symbols. Furthermore, map representations coming for instance from GIS (such as SE) are more limited regarding cartographic expressiveness, while focusing on minimalist but generic functionality. Point markers, lines, polygons, texts and raster images are integrated in the SE language but they are just used to change the basic symbology of maps served by a WMS. Its inappropriateness to create thematic maps is one of the main reasons why SLD-enabled WMS is used for presenting topographic maps [Sykora et al., 2007]. Furthermore, features expected by experienced cartographers such as advanced graphical primitives, patterns for spatial features, multi-level symbolization or masking are not present, thus limiting the use of WMS from a cartographic perspective.

Considering the cartographic representations of ArcGIS, their main goal is dedicated to improving the map clarity in ArcGIS products and not in providing a map description language. The cartographic representation in ArcGIS is in an undocumented and proprietary binary format and thus inherently linked with ArcGIS products such as ArcMap. Therefore, through their openness, the map representations (CartoWS, DiaML and SE) have a different focus than the cartographic representation of ArcGIS.

3.3.2 Analysis of Map Representations Regarding Flexibility

Regarding the flexibility aspect of a map representation, we are noticing though a slight trade-off between cartographic expressiveness and generic usage. Specific cartographic functions can be defined and referenced in a map representation, however they are linked to predefined datasets and to specific software. The cartographic representations of ArcGIS are the most obvious example since there are expressed in a proprietary format and only available for ArcGIS geodatabases. The syntax of CartoML allows map descriptions suitable for organized collections of statistic data. It introduces advanced cartographic features, but these symbolizations are adapted for the thematic data supplied by Eurostat's New Cronos REGIO database and rendered through the iMap visualization environment. DiaML significantly increases the cartographic expressiveness by allowing modeling of very complex point symbols, but it has to restrict the data format and the application domain in order to fit the logic for creating the symbols in the associated software. More precisely, CartoML leaves the cartographic expressiveness to the skilled programmer while DiaML replaces programming by geometric construction. On the other hand, SE is more generic in the sense that can be applied to any kind of spatial data, although it has more limited cartographic symbolization capabilities. This is understandable since it evolved to attend to the basic visualization requirements of GIS data presented through WMS, and it was not designed for cartography.

3.3.3 Conclusions of the Analysis

The analysis of the selected map representations was the basis for the further research towards a generic definition of a Cartographic Web Service. The analysis, that was performed mainly based on the evaluation of the cartographic expressivity and flexibility of the map representation, is summarized below:

Map representation	Human readability / Openness	Symbolization Expressivity	Flexibility / Generic usage with GIS data
CartoML	+	+	-
DiaML	+	+/- (*)	-
Symbology Encoding	+	+/- (**)	+
ArcGIS Representations	-	+	+/- (***)

* only considering point signatures

** in comparison with the other selected map representations

*** because of the limitation to ArcGIS geodatabases

The analysis of existing map representations offers a hint for answering the research questions of this doctoral thesis. Considering the examples of CartoML and SE, which cover thematic and topographic representations, we conclude that it is indeed **possible to formally describe the map symbolization.** Considering the example of SE/SLD we can also conclude that it is possible to have a **generic map representation**. The existence of the WMS standard proves the possibility for a **Web-based service-oriented** mechanism that is able to distribute maps on demand over the Web.

Therefore, when considering the cartographic functionality and expressivity required for development of cartographic Web services (abbreviated CartoWS), one can conclude that such services can offer a service-oriented mechanism similar to WMS, while the map creation should be based exclusively on a generic map representation that would combine the advanced cartographic output of CartoML and DiaML with the generality and flexibility of SE and SLD. Moreover, in this thesis we have considered an **approach based on already established standards** as a focus for the research in cartographic Web services. First of all, this approach is motivated by the conviction that related work performed by standardization bodies related to Web and geospatial standards are a good starting point and prevents "reinventing the wheel". Second, the existing mapping standards (WMS, SLD, SE) are flexible and they can be improved cartographically. Third, standards define an ordered and consistent manner of doing things and most importantly, it is a way of enabling interoperability. Finally, I believe that the cartographic improvement of the existing mapping standards is useful for the larger GIS community.

4

A Standards-based Map Representation for Cartography

This chapter defines the map representation, the core enabler of cartographic Web services. It describes a map-making process based on the SLD/SE standard. The focus lies on the enhancements brought to the SE standard for topographic and thematic symbolization, which effectively convert the well-known SLD/SE standards into map representation for cartography. The SE enhancements are formally defined in XML schema. The possibilities of the map representation syntax are also demonstrated with various visual exemplifications.

4.1 Considerations on a Map Representation

4.1.1 Building on Strong Foundations

From the analysis of the related work, we have established that in order to support the interoperability and the reuse of open standards, a cartographic web service (CartoWS) should be based on the WMS, SLD and SE. The main consideration was not to replace existing standards, but to enhance and extend them also for cartographic usage. We have concluded that CartoWS must offer a map representation that combines advanced cartographic output (e.g. CartoML) with the generality, flexibility and standardization of SLD and SE. Therefore, in CartoWS the map representation is contained in SLD, as the overall vehicle used for including the SE language. SE is the language for expressing the actual map representation independent of any service interface specification, and this research shows that it has the potential to express cartographic intentions and can be generalized and enhanced for professional cartographic usage. SE is defined by a FeatureTypeStyle, which contains several parameters, most import being rule definitions [OGC, 2005b].

On a conceptual level, we consider that a map representation needs three parts: **generic conditions** that specify what symbology to be applied considering the overall suitability for scale-dependent map symbolization, **specific conditions** for selecting and masking the geometric features to be symbolized, and finally the **definition of the symbology** itself.

4.1.2 Generic Conditions in a Map Representation

The main role of the generic conditions is to take into account the scale requirements for the visual differentiation of features or the legibility of the map. For example, let us consider we have to display many data points located over a small area in an overview map. When such sensors are displayed at small scales, the result is a form of "dot fever" when the density of point symbols confuses the map user and prevents the understanding of phenomena. Therefore, visualizations must be defined based on cartographic rules after careful consideration of scale and the density of information.

4.1.3 Specific Conditions and Symbology Definition

The specific conditions and the definition of the symbology represent the volume of the map representation. They will be introduced gradually through XML schema definitions and they will be graphically illustrated with a practical example of composing a topographic map based on a map representation [Iosifescu et al., 2010c]. The versatility of having a map representation, thus a descriptive and formal way of composing a map, is illustrated with the step-by-step construction starting from the most basic GIS data.

The examples will be reconstructing the Topographic Map of Methana mentioned in chapter 3, by following the original map compilation described in [Hurni, 1995]. The overall layer composition of the map will follow the traditional cartographic theory: the base layer represented by topography is followed by land cover, then by hydrology and glaciers, then by the anthropomorphic situation objects such as the transportation network and buildings, and finally ending with labels. The step-wise map definition and construction will serve in understanding the basic possibilities for a map representation for cartography, based on SLD/SE.

4.2 SE Foundations for a Map Representation for Cartography

4.2.1 The SE Rule

The foundation of map representation based on SE is represented by Rules. In the above considerations, we have established SE as the language for expressing the core map representation. SE Rules are an appropriate mechanism to express the cartographic intentions in a map representation since they group rendering instructions by feature-property conditions and map scales [OGC, 2005b].

The SE-based map representation is used to formally define the map output and is the fundamental requirement for defining automatic map generation from GIS data via a Web Service. In order to achieve the map representation the SE standard was extended with several important features that will be formally introduced using XML-Schema definitions and examples.

The following content requires the understanding of the SE standard [OGC, 2005b]. Selected SE components from this standard will be introduced as necessary in order to support the definitions of several cartographic elements necessary for the map representation.

An SE Rule is described by following XML-Schema fragment, which is extracted from the current Symbology Encoding standard:

```
<xsd:element name="Rule" type="se:RuleType">
</xsd:element>
<xsd:complexType name="RuleType">
   <xsd:sequence>
      <xsd:element ref="se:Name" minOccurs="0"/>
      <xsd:element ref="se:Description" minOccurs="0"/>
      <xsd:element ref="se:LegendGraphic" minOccurs="0"/>
      <xsd:element ref="se:MinScaleDenominator" minOccurs="0"/>
      <xsd:element ref="se:MaxScaleDenominator" minOccurs="0"/>
      <xsd:choice minOccurs="0">
         <xsd:element ref="ogc:Filter"/>
         <xsd:element ref="se:ElseFilter"/>
      </xsd:choice>
<xsd:element ref="se:Symbolizer" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

In the above SE Rule definition, it can be observed the main correspondences the basic elements present inside a Rule can be assigned to the conceptual classes discussed above: generic matching conditions (se:MinScaleDenominator, se:MaxScaleDenominator), specific matching conditions (ogc:Filter) and the symbology definition (se:Symbolizer).

The MinScaleDenominator and MaxScaleDenominator elements represent the minimum and maximum ranges of scale (denominators) of maps for which a rule should apply. The minimum scale is inclusive and the maximum scale is exclusive. The absence of a MinScaleDenominator element means there is no minimum-scale term to the condition, or that the default value is 0. The absence of a MaxScaleDenominator element means that there is no maximum-scale term to the condition, or that the default value is infinity. The definition is the same as in the SE standard.

As already mentioned, the scale denominators are important for multi-scale representation as illustrated in *Figure 8*.

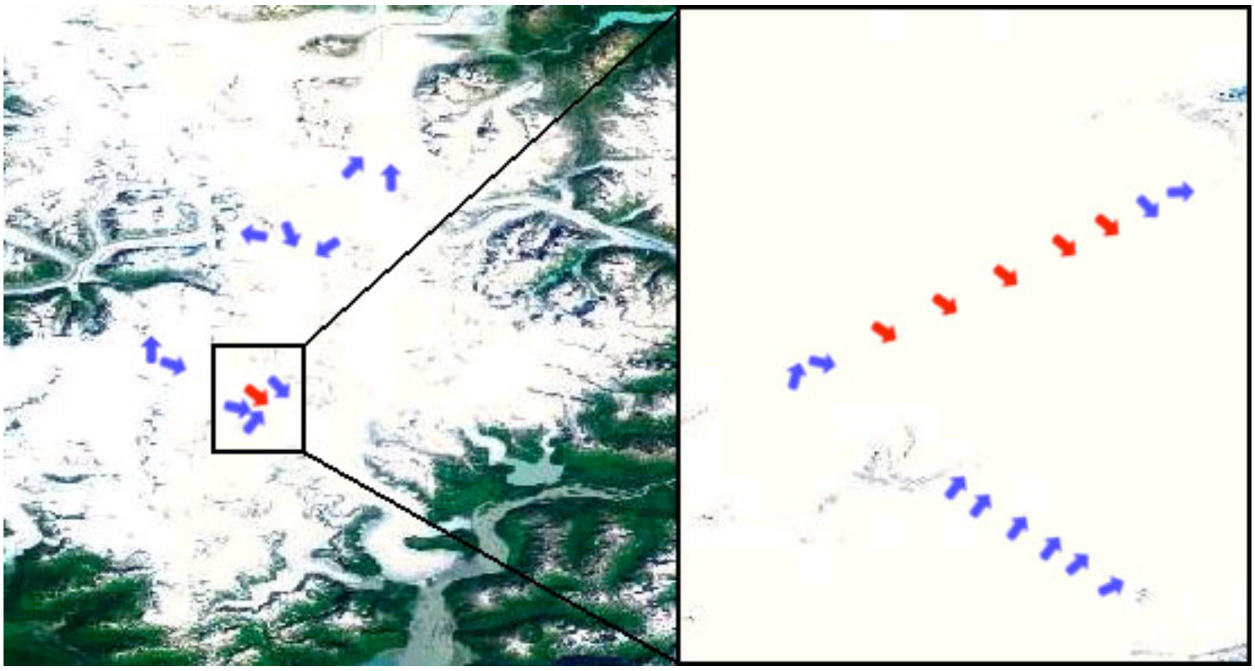


Figure 8. Example of multi-scale data visualization [Iosifescu et al., 2009]

In this example, simulated measurements of glacier movements are represented with arrows. The symbols are placed in the location of the measurements and rotated in the direction of the registered movement. Moreover, the measured values are again divided in two classes, with movements exceeding a certain threshold being represented in red.

The generic conditions define the selection of features that are presented in different scales. At small scales, only a few significant measurements are shown. If the user decides that some measurements require a closer look, more information is provided after zooming in the map. At larger scales - where overlapping does not occur - the number of presented features is increased. In this manner, the user receives the information in a legible manner starting from the overview of the region. Furthermore, we recommend that any cartographic software interpreting the scale denominators is aware of the resolution of the output device used for presenting the rendered map at a given scale. Via a dots-per-inch (DPI) parameter added to a map rendering operation, the values used for the scale denominators can be made more generic than just for the display on contemporary video displays. If a DPI is specified, than the scale calculation follows the known algorithm; for example, the computation for DPI = 100 is: 100 dpi => 1/100 inches × 25.4 mm/inch = 0.254 mm (pixel size) => 0.254 mm × 1000 mm/m = 0.000254 m (pixel size in meters). In the absence of the DPI information, the scale is computed with the standardized rendering pixel size of 0.28mm [ISO, 2005].

4.2.2 The Filter

The Filter element of a Rule allows a detailed expression and selection of features based on the OGC Filter Encoding Standard [OGC, 2002a]. The following XML schema fragment describes the Filter:

```
 <xsd:complexType name="FilterType">
   <xsd:choice>
      <xsd:element ref="ogc:spatialOps"/>
      <xsd:element ref="ogc:comparisonOps"/>
      <xsd:element ref="ogc:logicOps"/>
      <xsd:element ref="ogc:_Id" maxOccurs="unbounded"/>
   </xsd:choice>
</xsd:complexType>
```

From the Filter schema presented above we notice three types of operators that can be jointly used in order to assign a symbol to a certain spatial feature: comparison, logic and spatial operators.

4.2.3 Comparison Operators

Comparison operators are used to form boolean expressions that evaluate the mathematical comparison between two arguments. Their semantics are described in detail in the OGC Filter Encoding Specifications. The available comparison operators are visible in the following schema:

```
<xsd:element name="PropertyIsEqualTo"
   type="ogc:BinaryComparisonOpType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsNotEqualTo"
   type="ogc:BinaryComparisonOpType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsLessThan"
   type="ogc:BinaryComparisonOpType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsGreaterThan"
   type="ogc:BinaryComparisonOpType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsLessThanOrEqualTo"
   type="ogc:BinaryComparisonOpType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsGreaterThanOrEqualTo"
   type="ogc:BinaryComparisonOpType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsLike"
   type="ogc:PropertyIsLikeType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsNull"
   type="ogc:PropertyIsNullType" substitutionGroup="ogc:comparisonOps"/>
<xsd:element name="PropertyIsBetween"
   type="ogc:PropertyIsBetweenType" substitutionGroup="ogc:comparisonOps"/>
```

The purpose of the filters can be illustrated with one of the first steps in the reconstruction of the topographic map of Methana. This step is represented by the definition of the map representation for land cover. As starting point we know that the specific land cover dataset for Methana contains polygon geometries with the following attached attributes: id, VEGE_TYPE, Area. The attribute VEGE_TYPE can be used for discriminating between the following classes: barren_land, cultivated_area, forest, rock, scrub, settlement. These classes can be symbolized with the following information:

- barren_land: there is no polygon stroke color, polygon fill color is rgb(253, 255, 228).
- cultivated_area: there is no polygon stroke color , polygon fill color is rgb(118, 170, 22).
- forest: polygon fill color is rgb(3, 127, 21), polygon stroke color is rgb(3, 255, 29).
- rock: there is no polygon stroke color , polygon fill color is rgb(176, 197, 214).
- scrub: there is no polygon stroke color, polygon fill color is rgb(255, 255, 127).
- settlement: polygon fill color is rgb(131, 41, 0), polygon stroke color rgb(230, 37, 66).

A fragment of the map representation defined for the barren_land category would therefore make use of the “PropertyIsEqualTo” comparison operator as follows:

```
<Rule xmlns="http://www.opengis.net/sld">
   <Filter xmlns="http://www.opengis.net/ogc">
      <PropertyIsEqualTo xmlns="http://www.opengis.net/ogc">
         <PropertyName xmlns="http://www.opengis.net/ogc">VEGE_TYPE</PropertyName>
         <Literal xmlns="http://www.opengis.net/ogc">barren_land</Literal>
      </PropertyIsEqualTo>
   </Filter>
   <PolygonSymbolizer xmlns="http://www.opengis.net/sld">
      <Fill xmlns="http://www.opengis.net/sld">
         <CssParameter xmlns="http://www.opengis.net/sld" name="fill" >#fdffe4</CssParameter>
      </Fill>
   </PolygonSymbolizer>
</Rule>
```

Similar SE rules must be defined for the map representation of the remaining classes. Corresponding to the characteristics for land cover, comparison operators can be used to assign a specific style to each land cover category as shown in the next figure (the visualization from *Figure 9* is obtained when the whole land cover layer is made approximately 33% transparent and overlaid on a shaded relief of the area).

The remaining comparison operators (e.g. PropertyIsGreaterThan, PropertyIsBetween, etc…) can be used in a similar manner to create conditional expressions in order to assign a specific style to a selected category of geospatial features.

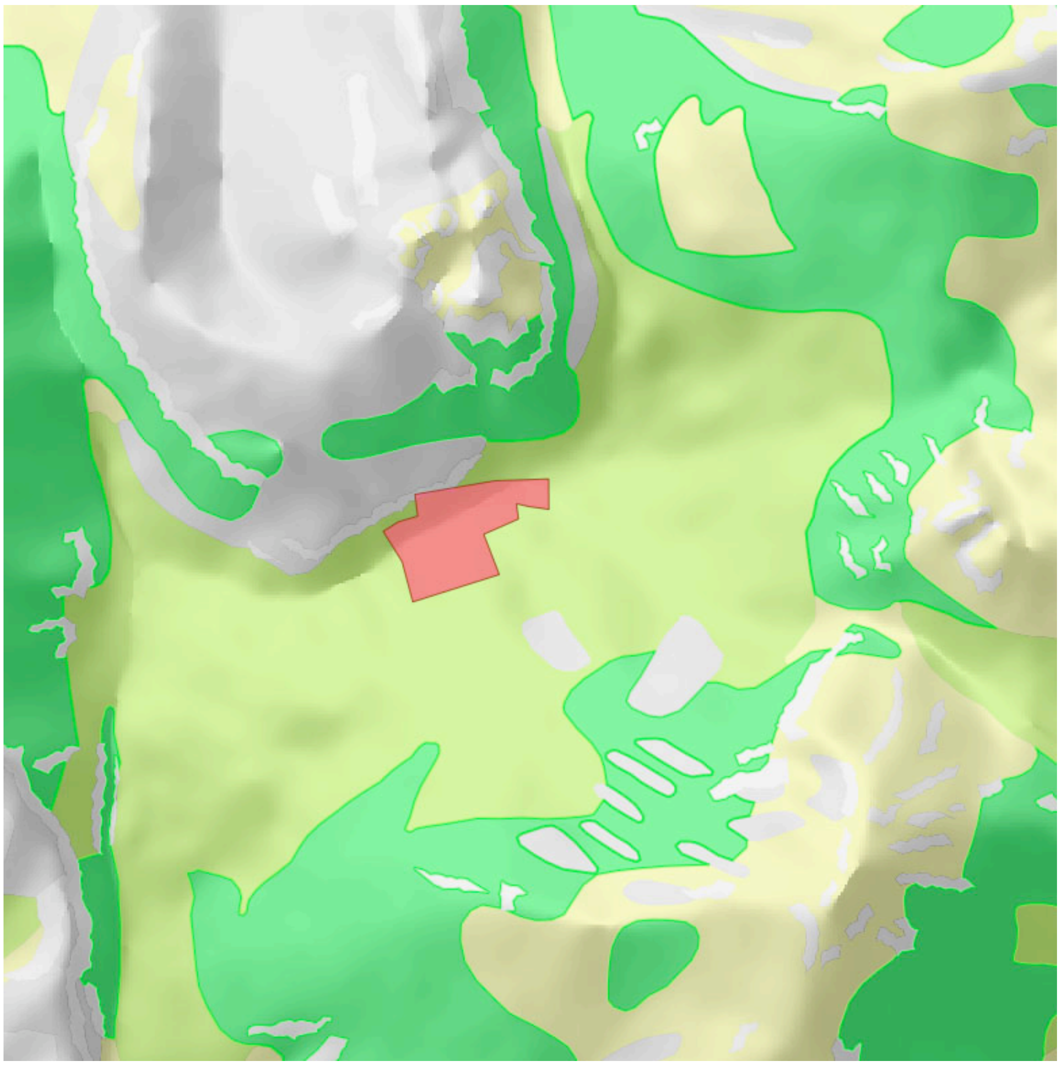


Figure 9. Symbolization for land cover with comparison operators

4.2.4 Spatial Operators for Features Selection

Spatial operators as defined by the OGC Filter Encoding Specifications support the testing of spatial relationships between two geometric objects. They are Boolean methods that are used to test for the existence of a specified topological spatial relationship between spatial features, as they would be represented on a map. The existing OGC spatial operators are shown formally in the following schema fragment:

```
 <xsd:element name="Equals"
   type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Disjoint"
   type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Touches"
   type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Within"
   type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Overlaps"
   type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Crosses"
   type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Intersects"
   type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
```

```
<xsd:element name="Contains"
   type="ogc:BinarySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="DWithin"
   type="ogc:DistanceBufferType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Beyond"
   type="ogc:DistanceBufferType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="BBOX"
   type="ogc:BBOXType" substitutionGroup="ogc:spatialOps"/>
```

The detailed definition and semantics of these Boolean spatial operators can be found in the OGC Filter Encoding Standard [OGC, 2002a], to which this thesis adheres. They also serve to create conditional expressions in order to assign a specific style to a category of geospatial features, selected based on spatial relationships.

4.2.5 Logical Operators

Logical operators are used to combine one or more conditional expressions. Their semantic is detailed also in the OGC Filter Encoding specifications. The next XML schema fragment defines the logical operators and their use for a more complex interweaving of the comparison and spatial operators:

```
<xsd:element name="And" type="ogc:BinaryLogicOpType" substitutionGroup="ogc:logicOps"/>
<xsd:element name="Or" type="ogc:BinaryLogicOpType" substitutionGroup="ogc:logicOps"/>
<xsd:element name="Not" type="ogc:UnaryLogicOpType" substitutionGroup="ogc:logicOps"/>
<xsd:element name="logicOps" type="ogc:LogicOpsType" abstract="true"/>
<xsd:complexType name="LogicOpsType" abstract="true"/>
<xsd:complexType name="BinaryLogicOpType">
   <xsd:complexContent>
      <xsd:extension base="ogc:LogicOpsType">
         <xsd:choice minOccurs="2" maxOccurs="unbounded">
            <xsd:element ref="ogc:comparisonOps"/>
            <xsd:element ref="ogc:spatialOps"/>
            <xsd:element ref="ogc:logicOps"/>
         </xsd:choice>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="UnaryLogicOpType">
   <xsd:complexContent>
      <xsd:extension base="ogc:LogicOpsType">
         <xsd:sequence>
            <xsd:choice>
               <xsd:element ref="ogc:comparisonOps"/>
               <xsd:element ref="ogc:spatialOps"/>
               <xsd:element ref="ogc:logicOps"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

Illustrations that show the nesting of various conditions inside logical operators for the assignment of complex styling can be found in the examples presented in the next subchapter.

4.3 Descriptive Geometry Processing for Cartographic Purposes

4.3.1 Extension of Spatial Operators for Masking

In the framework of this work, the map description was defined having enough cartographic expressivity in order to enable cartographers to describe complex symbolization behaviors. This is an important issue, since cartographers are usually requiring complex geometry manipulations for masking purposes while the current usage of the Filter only for the selection of features is inherently restrictive. Therefore, this work enhances the use of the spatial operators with manipulation of geometries to enable functionalities similar to cartographic masking. As a result, the expressivity of the map representation goes beyond the simple feature selection controlled by attribute conditions present in the current SLD/SE standards, while including and therefore keeping the conformance with the OGC Filter Encoding Specifications. The Filter is extended with powerful spatial operators that not only express cartographic selections but also the manipulation of geometries.

4.3.2 Geometry Spatial Operators

Another OGC standard, the Simple Features Implementation Specification has all the concepts required to enable the descriptive modeling of cartographic manipulation of geometries as long as the relationship between symbolized features and geometry is enforced.

Therefore, the Binary Spatial Operators presented above can be extended with Geometry Spatial Operators as presented formally with the XML schema fragments. First, the new Geometry Spatial Operators XML type is defined:

```
 <xsd:complexType name="GeometrySpatialOpType">
   <xsd:complexContent>
     <xsd:extension base="ogc:SpatialOpsType">
       <xsd:sequence>
         <xsd:choice>
           <xsd:element ref="gml:Geometry"/>
           <xsd:element ref="gml:Envelope"/>
           <xsd:element ref="sld:NamedLayer"/>
           <xsd:element ref="sld:UserLayer"/>
         </xsd:choice>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

Based on the above, the following well-known geometry manipulation operators can be defined:

```
<xsd:element name="Intersection"
type="ogc:GeometrySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Union"
type="ogc:GeometrySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Difference"
type="ogc:GeometrySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="SymDifference"
type="ogc:GeometrySpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:element name="Distance"
type="ogc:GeometrySpatialOpType" substitutionGroup="ogc:spatialOps"/>
```

The semantics of the above operators are described in detail in the Simple Features Implementation Specifications. The effect of several geometry spatial operators is illustrated in JTS Technical Specification [Vivid Solutions, 2010] after which we created the next figure:

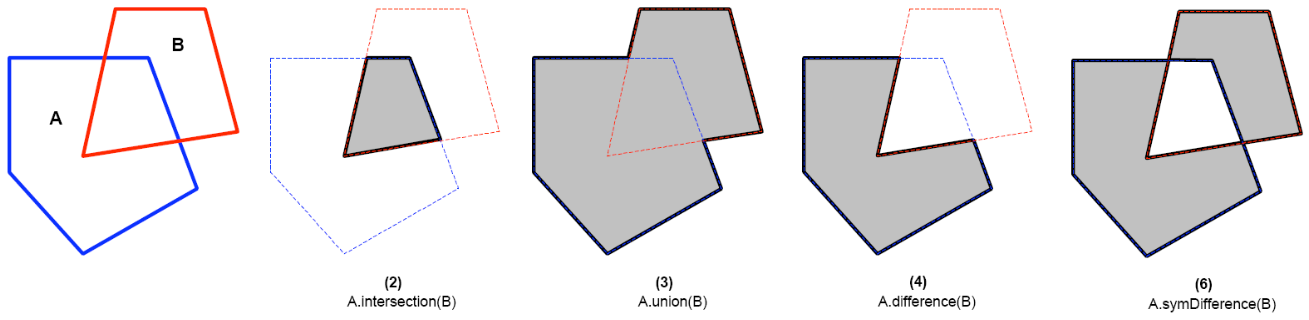


Figure 10. Effects of spatial operators on polygon geometries [after Vivid Solutions, 2010]

For layer masking purposes, the remaining spatial methods defined in the Simple Features Implementation Specifications for creating buffers and a convex hull are also introduced. For the sake of completeness they are formally defined as follows:

```
<xsd:element name="Buffer"
type="ogc:BufferSpatialOpType" substitutionGroup="ogc:spatialOps"/>
<xsd:complexType name="Buffer SpatialOpType">
   <xsd:complexContent>
      <xsd:extension base="ogc:SpatialOpsType">
         <xsd:sequence>
            <xsd:element name="Distance" type="ogc:DistanceType"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="ConvexHull"  type="xsd:any" substitutionGroup="ogc:spatialOps"/>
```

Conceptually, all Geometry Spatial Operators apply to the immediate geometry definition, either the layer itself or to the geometry above in the nesting of a complex expression. Most Geometry Spatial Operators take as input a geometry and provide as output a geometry that can be used in every expression that expects a gml:Geometry or gml:Envelope operator. Similarly, the Distance operator can be used in place of an ogc:Literal in other more complex expressions.

4.3.3 Usage of Geometry Spatial Operators for Masking

The usefulness of spatial operators is shown with another step from the reconstruction of the topographic map of Methana. In this step we would like to integrate the contours in the map of Methana based on map representation.

The contours are available as separate polyline geometries with the following attached attributes: id, ELEV, TYPE. In the printed map of Methana, contours are classified according to their ELEV (elevation) and TYPE. Major contours are having the elevation divided by 100, intermediate contours by 20, and the remaining contours are supplementary. These classes are symbolized with the following style:

- index: polyline stroke size is 2, continuous line style, polyline stroke color is rgb(170,85,0).

- intermediary: polyline stroke size is 1, continuous line style, polyline stroke color is rgb(170,85,0).

- supplementary: polyline stroke size is 1, dash line pattern consisting of short lines separated by spaces instead of a continuous line, and polyline stroke color is rgb(170,85,0).

Since we know their symbolization, we might simply consider overlaying the height contours on top of the land cover. However, from [Hurni, 1995] we note that contours should not be displayed on top of cliffs due to the high variance in height. Moreover, in order to improve the visual contrast of the contours and consequently the map legibility, [Hurni, 1995] assigns a different symbolization for the contours on top of barren land than for the remaining relevant contours. Therefore, the contours on top of barren_land areas of land cover are given the polyline stroke color of rgb(0,0,0) as shown in *Figure 11*.

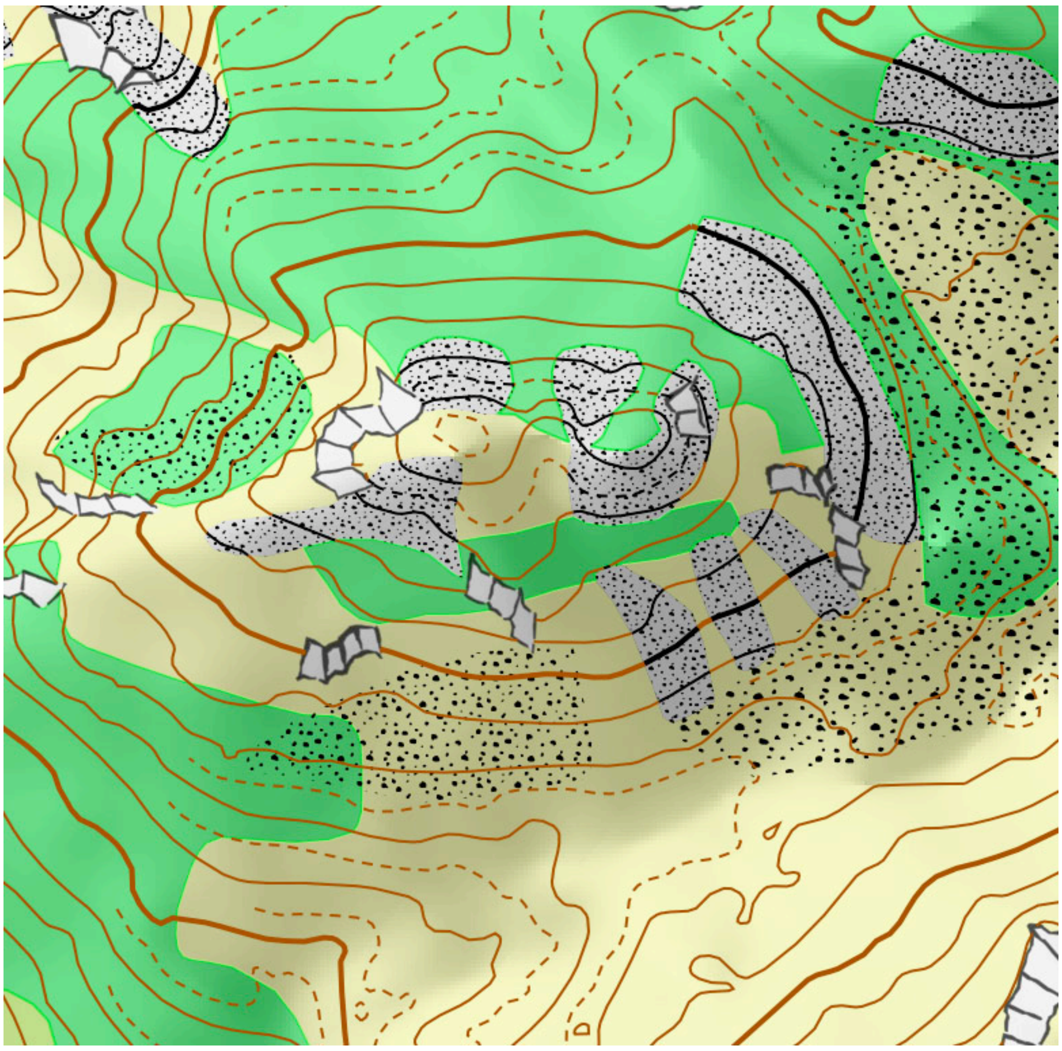


Figure 11. Symbolization of land cover and contours

In order to express this specific Filter for contours, first we must extract only the contours that do not intersect areas covered by cliffs. For this purpose, we can make use of the defined Difference spatial operator, which eliminates the contours on top of the cliff areas. Then we can think of combining the Difference spatial operator with the Intersection operator in order to select only the relevant contours according to our cartographic requirements. The combination is achieved using a logical operator mentioned in section 4.2.5, in order to address only the contours geometries that simultaneously result from the Difference and the Intersection operators.

With the above operators we can construct a Filter that defines that the features present in the contours layer are to be clipped with the rock areas and in addition to be intersected with the barren_land areas as shown below:

```
<Filter xmlns="http://www.opengis.net/ogc">
  <And>
    <Difference>
      <NamedLayer xmlns="http://www.opengis.net/sld">
        <Name xmlns="http://www.opengis.net/sld">Vegetation</Name>
        <UserStyle xmlns="http://www.opengis.net/sld">
          <FeatureTypeStyle xmlns="http://www.opengis.net/sld">
            <Rule xmlns="http://www.opengis.net/sld">
              <Filter xmlns="http://www.opengis.net/ogc">
                <PropertyIsEqualTo xmlns="http://www.opengis.net/ogc">
                  <PropertyName xmlns="http://www.opengis.net/ogc">VEGE_TYPE</PropertyName>
                  <Literal xmlns="http://www.opengis.net/ogc">rock</Literal>
                </PropertyIsEqualTo>
              </Filter>
          </FeatureTypeStyle>
        </UserStyle>
      </NamedLayer>
    </Difference>
    <Intersection>
      <NamedLayer xmlns="http://www.opengis.net/sld">
        <Name xmlns="http://www.opengis.net/sld">Vegetation</Name>
        <UserStyle xmlns="http://www.opengis.net/sld">
          <FeatureTypeStyle xmlns="http://www.opengis.net/sld">
            <Rule xmlns="http://www.opengis.net/sld">
              <Filter xmlns="http://www.opengis.net/ogc">
                <PropertyIsEqualTo xmlns="http://www.opengis.net/ogc">
                  <PropertyName xmlns="http://www.opengis.net/ogc">VEGE_TYPE</PropertyName>
                  <Literal xmlns="http://www.opengis.net/ogc">barren_land</Literal>
                </PropertyIsEqualTo>
              </Filter>
          </FeatureTypeStyle>
        </UserStyle>
      </NamedLayer>
    </Intersection>
  </And>
</Filter>
```

The defined comparison, logical and spatial operators can be used to build complex map representations by nesting the operators as described in the Filter Encoding Specifications [OGC, 2002a, 2009]. The more complex the visualization is, the more complex description it produces. For example, contours must also be symbolized according to their elevation and type (index, intermediary, supplementary) and comparison conditions can be further nested with additional logical operators to enable the desired effect:

```
<Rule xmlns="http://www.opengis.net/sld">
  <Filter xmlns="http://www.opengis.net/ogc">
    <And>
      <And>
        <Difference>
          <NamedLayer xmlns="http://www.opengis.net/sld">
            <Name xmlns="http://www.opengis.net/sld">Vegetation</Name>
            <UserStyle xmlns="http://www.opengis.net/sld">
```

```
                    <FeatureTypeStyle xmlns="http://www.opengis.net/sld">
                      <Rule xmlns="http://www.opengis.net/sld">
                        <Filter xmlns="http://www.opengis.net/ogc">
                          <PropertyIsEqualTo xmlns="http://www.opengis.net/ogc">
                            <PropertyName xmlns="http://www.opengis.net/ogc">VEGE_TYPE</PropertyName>
                            <Literal xmlns="http://www.opengis.net/ogc">rock</Literal>
                          </PropertyIsEqualTo>
                        </Filter>
                    </FeatureTypeStyle>
                  </UserStyle>
                </NamedLayer>
              </Difference>
              <Intersection>
                <NamedLayer xmlns="http://www.opengis.net/sld">
                  <Name xmlns="http://www.opengis.net/sld">Vegetation</Name>
                  <UserStyle xmlns="http://www.opengis.net/sld">
                    <FeatureTypeStyle xmlns="http://www.opengis.net/sld">
                      <Rule xmlns="http://www.opengis.net/sld">
                        <Filter xmlns="http://www.opengis.net/ogc">
                          <PropertyIsEqualTo xmlns="http://www.opengis.net/ogc">
                            <PropertyName xmlns="http://www.opengis.net/ogc">VEGE_TYPE</PropertyName>
                            <Literal xmlns="http://www.opengis.net/ogc">barren_land</Literal>
                          </PropertyIsEqualTo>
                        </Filter>
                    </FeatureTypeStyle>
                  </UserStyle>
                </NamedLayer>
              </Intersection>
            </And>
            <PropertyIsEqualTo xmlns="http://www.opengis.net/ogc">
              <PropertyName xmlns="http://www.opengis.net/ogc">TYPE</PropertyName>
              <Literal xmlns="http://www.opengis.net/ogc">index</Literal>
            </PropertyIsEqualTo>
          </And>
        </Filter>
        <LineSymbolizer xmlns="http://www.opengis.net/sld">
          <Stroke xmlns="http://www.opengis.net/sld">
            <CssParameter xmlns="http://www.opengis.net/sld" name="stroke" >#000000</CssParameter>
            <CssParameter xmlns="http://www.opengis.net/sld" name="stroke-width" >2</CssParameter>
          </Stroke>
        </LineSymbolizer>
      </Rule>
```

As a result of this Filter, the desired masking of features is obtained dynamically, without the need for manual processing or intermediate datasets. Moreover, the Geometry Spatial Operators introduced in this thesis also alleviates the need for cartographic tricks such as multiple overlays of differently rendered contours in order to obtain a similar effect. With a well-defined map representation, the contours are always rendered as a single logical layer.

4.3.4 Additional Considerations on the Usage of Geometry Spatial Operators

The more complex possibilities for masking can be understood with the syntax of the GeometrySpatialOpType. From the XML-schema fragment presented in 4.3.2, it can be noticed that it allows the use of symbolized layers in Geometry Spatial Operators. When the parameterized geometry is defined by the sld:NamedLayer or the sld:UserLayer elements, it is possible to include the reference to an sld:NamedStyle or sld:UserStyle. In such cases, the geometry of the symbolized features referred by the sld:NamedLayer or the sld:UserLayer elements is used in expressions instead of the simple feature geometry. For example, in a Difference operator as previously illustrated, the A geometry is represented by the input layer to which additional SE rules directly applies, and the B geometry is represented by another layer (symbolized or not) as referenced by sld:NamedLayer or sld:UserLayer elements.

Moreover, for cartographers that want complete control over the extraction of features for symbolization, it is also possible to use buffers in order to preemptively simulate the effect of the symbolization on spatial features before applying additional Geometry Spatial Operators. The advanced use of the proposed geometry spatial operators not only provides cartographers with the expressiveness and flexibility required for the creation of map representations but it is envisioned that this extension will play an important role in the detection and elimination of cartographic conflicts.

The extended spatial operators enhance the expressiveness of the Filter beyond the definition of a combination of one or more symbols that evaluate to a single boolean value of true or false. The enhanced OGC Filter can now be used not only to select features from the original layer, but also to transiently modify the features according to cartographic requirements.

All geometry processing spatial operators were defined with the sole goal of improving the expressiveness of the symbolization by geometrically extracting the right features for the map from the bulk of source GIS data. With the presented syntax, we are now able to access the data source geometries and to process them in a descriptive way for styling purposes. It must be mentioned that these geometry manipulations do not affect the original data in any way and that the obtained masks/features are not persistent in the source data.

4.4 Symbology definitions in the Map Representation

4.4.1 Understanding the Symbolization Possibilities of the Map Representation

The final ingredients of the map representation are the Symbolizers, which describe how geometries are to be represented on the map. The geometries to be symbolized are either original features matching a selection or transient geometries obtained using the Geometry Spatial Operators.

The underlying model of rendering and map composition is controlled by the sequence of matching SE Rules. The symbolizers are applied following the painter's model [W3C, 2003]. This model allows generating complex map signatures with several "layers" of symbols controlled by the sequence of FeatureTypeStyle elements.

The expressiveness of the symbolization is one of the major concerns in cartography. We will demonstrate that cartographic symbolization can be achieved starting from the following symbolizers: Polygon, Line, Point, Text, Raster and Diagram Symbolizers. Beside the completely new Diagram Symbolizer, all the other symbolizers are inherited from SE in order to achieve a complete compatibility with the existing standards. However, while the SE Symbolizers and their syntax are inherited, they are also enhanced in order to achieve the goal of providing expressive means for map symbolization. The capabilities of the Symbolizers are to be explained considering their cartographic expressiveness and the main extensions added to the SE standard are going to be highlighted.

4.4.2 Symbolization of Areas with Patterns and Gradients

Vectors represent the focus of the symbolization process. Unlike the usually straightforward symbolization for raster data, the symbolization of polygon, line and point geometries may require advanced graphic representation.

The description for the symbolization of polygons is performed with the PolygonSymbolizer, defined formally by the SE standard as:

```
<xsd:element name="PolygonSymbolizer" type="se:PolygonSymbolizerType" substitutionGroup="se:Symbolizer"/>
<xsd:complexType name="PolygonSymbolizerType">
  <xsd:complexContent>
    <xsd:extension base="se:SymbolizerType">
      <xsd:sequence>
        <xsd:element ref="se:Geometry" minOccurs="0"/>
        <xsd:element ref="se:Fill" minOccurs="0"/>
        <xsd:element ref="se:Stroke" minOccurs="0"/>
        <xsd:element ref="se:Displacement" minOccurs="0"/>
        <xsd:element ref="se:PerpendicularOffset" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The PolygonSymbolizer, as defined in the SE standard, mostly covers the basic representation of polygon geometries. The Fill element specifies well how the area of the geometry will be colored. On the other hand, the Fill element of the PolygonSymbolizer is extended in a CartoWS with encodings for patterns and gradients for polygons or area-type geometries. Patterns and gradients are normally encountered in a variety of cartographic representations; therefore a vector-based solution for expressing patterns is an important addition brought to the SE standard in this thesis.

The definition of the patterns and gradients are an extension to the SE standard for cartography. They are defined based on the widely accepted standard for Scalable Vector Graphics [W3C, 2003]. Moreover, SVG has been successfully applied in cartography due to its superior graphic description capabilities [Neumann, 2005].

The SE already uses several SVG/CSS2 parameters with identical names and semantics, thus inclusion of further SVG elements is a straightforward process. The chosen solution for the definition of patterns and gradients is to enhance the Fill element with the corresponding SVG elements. Therefore the Fill element is extended as shown below:

```
<xsd:element name="Fill" type="se:FillType"/>
<xsd:complexType name="FillType">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element ref="se:GraphicFill" minOccurs="0"/>
         <xsd:element ref="se:SvgParameter" minOccurs="0" maxOccurs="unbounded"/>
         <xsd:element ref="se:SvgRadialGradient" minOccurs="0"/>
         <xsd:element ref="se:SvgLinearGradient" minOccurs="0"/>
         <xsd:element ref="se:SvgPattern" minOccurs="0"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

The SvgPattern, SvgRadialGradient and SvgLinearGradient elements must contain code that defines a pattern, respectively a gradient, in valid SVG syntax. Therefore these elements are defined formally with the corresponding SVG types:

```
<xsd:element name="SvgRadialGradient" type="svg:radialGradient"/>
<xsd:element name=" SvgLinearGradient" type="svg:linearGradient"/>
<xsd:element name=" SvgPattern " type="svg:pattern"/>
```

Currently no ConicalGradient is defined and future extensions may choose to include one.

The syntax allows raster images to be embedded in the SVG definition for a pattern (texture mapping), although this practice is discouraged due to the fact that raster images are not scalable.

The possibility to define patterns for polygon geometries is an important feature especially for the natural hazards maps. The SE extension allows to create fill patterns with SVG that are arbitrarily complex as shown in *Figure 12*, and it can be used to define even hatch patterns as proven in [Eberle, 2009].

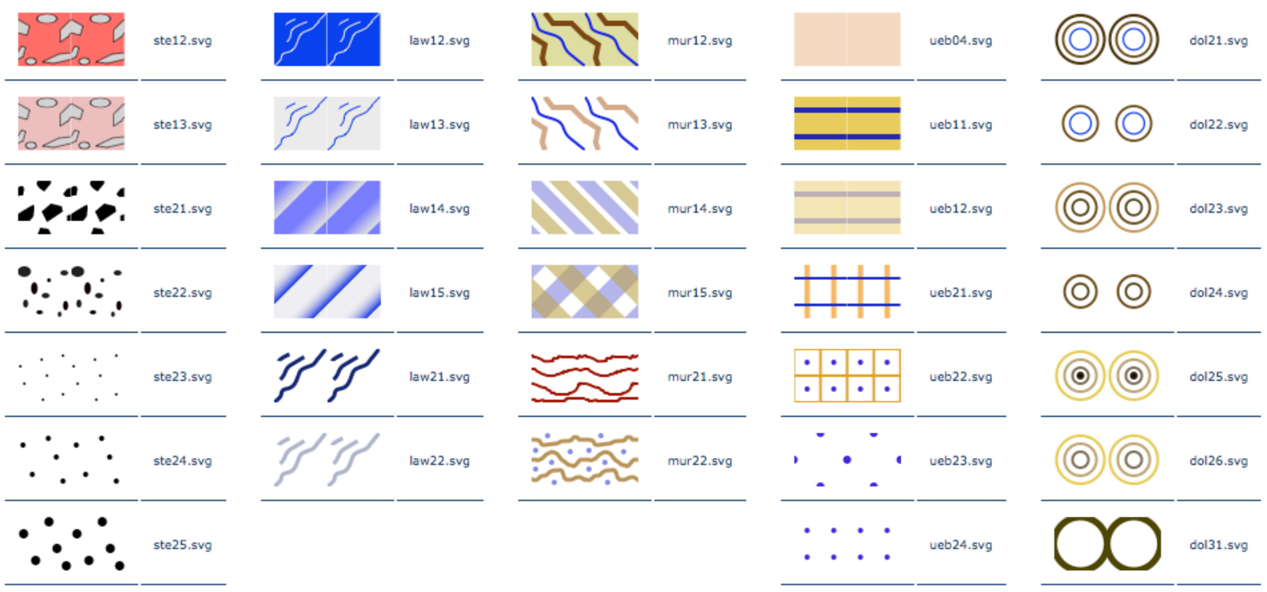


Figure 12. Examples of SVG fill patterns usable in a map representation [after Eberle, 2009]

Furthermore, there is no restriction applied to the complexity of the patterns regarding their integration in the map representation.

4.4.3 The Usefulness of Area Symbolization with Patterns

The usefulness of patterns can be illustrated with the example of scree representation in the topographic map of Methana. The example will enhance the situation presented in *Figure 9* with scree areas. Scree areas are polygon geometries textured with patterns consisting of small points with various dimensions and arrangements [Hurni, 1995]. We want to show that such patterns can be expressed in SVG and included in a map representation.

We start with a simple pattern definition composed from a single circle as shown below:

```
<pattern width="30" height="30" x="0" y="0">
            <g xmlns="http://www.w3.org/2000/svg">
                        <circle cx="5" cy="5" r="2" fill="#000000"/>
            </g>
</pattern>
```

This pattern can be used as a Fill for scree polygons as shown below:

```
<FeatureTypeStyle xmlns="http://www.opengis.net/sld">
  <Rule xmlns="http://www.opengis.net/sld">
    <Filter xmlns="http://www.opengis.net/ogc">
      <PropertyIsEqualTo xmlns="http://www.opengis.net/ogc">
        <PropertyName xmlns="http://www.opengis.net/ogc">SCREE_TYPE</PropertyName>
        <Literal xmlns="http://www.opengis.net/ogc">scree_1</Literal>
      </PropertyIsEqualTo>
    </Filter>
    <PolygonSymbolizer xmlns="http://www.opengis.net/sld">
      <Fill>
        <pattern width="30" height="30" x="0" y="0">
          <g xmlns="http://www.w3.org/2000/svg">
            <circle cx="5" cy="5" r="2" fill="#000000"/>
          </g>
        </pattern>
      </Fill>
    </PolygonSymbolizer>
  </Rule>
```

```
</FeatureTypeStyle>
```

In addition, for different types of scree we can use pattern definitions with slightly different radius of the circle. With such a map representation, scree areas are filled using the circle patterns, which results in the repetitive symbolization of scree areas visible in Figure 13.

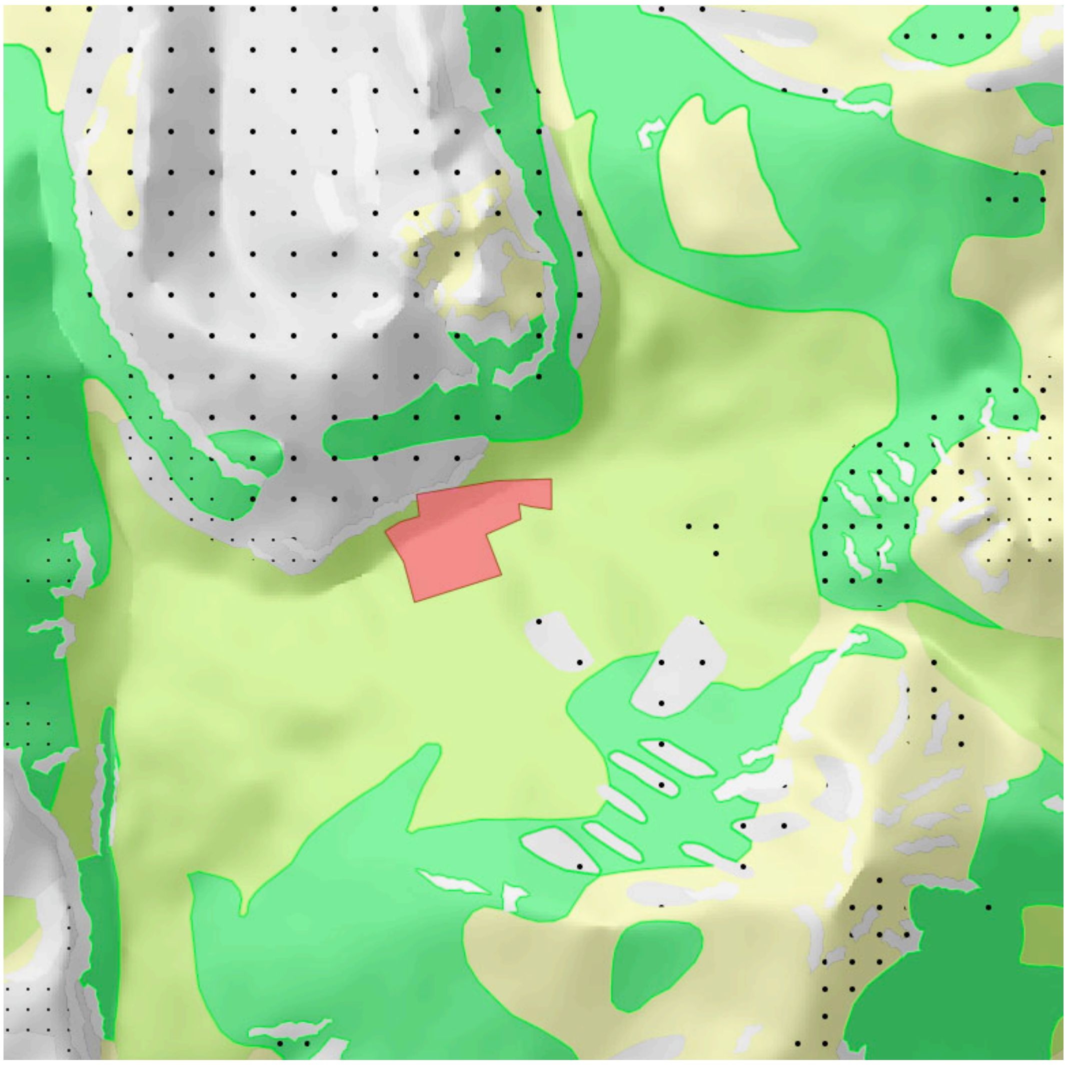


Figure 13. Repetitive patterns for scree areas

The example of the simple screen pattern is only provided in order to understand the mechanism of pattern symbolization since the result is unsatisfactory from the cartographic point of view due to the regular distribution of the points that compose the patterns. However, as shown below more complex SVG elements can be used to define patterns according to the cartographic requirements for different types of scree, with the visual representation shown in Figure 14.

```
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
x="0px" y="0px" width="323.14px" height="267.315px" viewBox="0 0 323.14 267.315" enable-background="new 0 0
323.14 267.315" xml:space="preserve">
                <path d="M9.569,4.255C-1.364,4.924-
                2.399,28.672,3.782,34.616c5.647,0.33,14.727,1.745,20.194,0.147c5.896-1.723,8.137-8.03,9.319-
                13.562C36.612,5.672,23.998-2.417,10.417,3.407"/>
                <path
                d="M120.586,21.204C113.793,4.966,84.928,9.415,90.925,31.374c1.894,6.934,12.141,8.894,20.336,6.7
                77c9.892-2.555,7.63-11.346,7.63-20.336"/>
                <path d="M151.942,15.272c-5.958,7.038-9.222,11.964-
                8.327,21.889c7.083,6.904,29.697,3.906,39.555,3.407
                c0.418-8.245,2.216-17.735-5.993-22.582c-6.386-3.77-16.745-2.714-24.388-2.714"/>
                <path d="M238.382,0.018c-1.798-0.159-3.934,0.819-5.804,0.976c-0.298,5.831-
                1.657,11.553,4.066,14.323c3.689,1.786,13.587,0.037,17.712-0.172c0.69-13.617-1.821-15.08-15.126-
                15.126"/>
               <!-- further path definitions were omitted  -->
</svg>
```

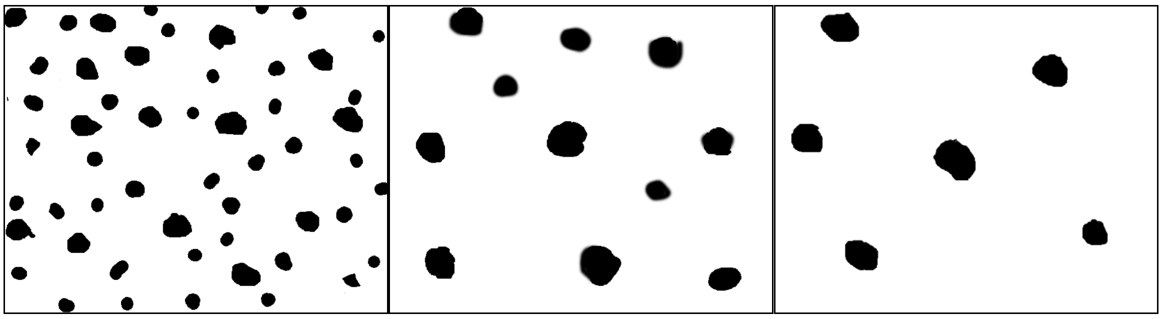


Figure 14. Patterns defined for different types of scree

The straightforward ways for creating complex patterns is to use vector graphic programs like Inkscape (open source) or Adobe Illustrator (commercial) and then export the SVG code. All the graphical possibilities offered by the programs can be used without restrictions (including patterns, gradients, Bezier curves).

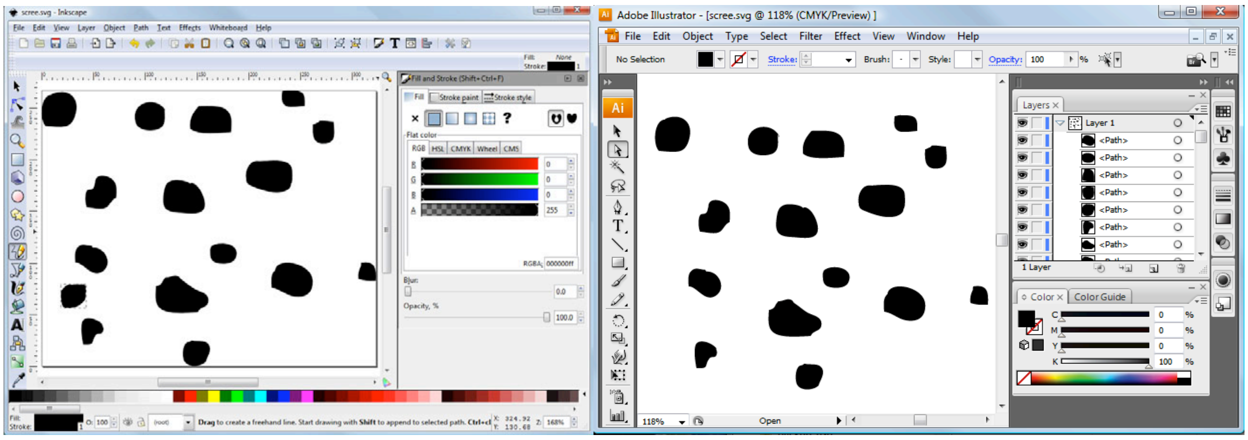

scree.svg - Inkscape
Adobe Illustrator - [scree.svg @ 118% (CMYK/Preview)]
Fill and Stroke (Shift+Ctrl+F)
Layers
Layer 1
<Path>
Color Guide

Figure 15. Creation of cartographic patterns in vector graphics software

After creating the patterns according to cartographic requirements, such drawings can be exported as SVG and used in a PolygonSymbolizer:

```
  <PolygonSymbolizer xmlns="http://www.opengis.net/sld">
      <Fill>
         <pattern width="50" height="50" x="0" y="0">
                     <svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg"
          xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" width="323.14px" height="267.315px"
          viewBox="0 0 323.14 267.315" enable-background="new 0 0 323.14 267.315" xml:space="preserve">
                              <path d="M9.569,4.255C-1.364,4.924-
                     2.399,28.672,3.782,34.616c5.647,0.33,14.727,1.745,20.194,0.147c5.896-1.723,8.137-8.03,9.319-
                     13.562C36.612,5.672,23.998-2.417,10.417,3.407"/>
                              <path
                     d="M120.586,21.204C113.793,4.966,84.928,9.415,90.925,31.374c1.894,6.934,12.141,8.894,20.336,6
                     .777c9.892-2.555,7.63-11.346,7.63-20.336"/><!-- further path definitions were omitted  -->
                     </svg>
         </pattern>
      </Fill>
   </PolygonSymbolizer>
```

As a consequence, the complexity of the defined patterns can be increased according to cartographic requirements without restrictions. Even though a pattern is never random and there is no possibility for defining irregular patterns such as described in [Jenny et al., 2010], the more complex the pattern definition is, the more it alleviates or even eliminates the perception of pattern repetition as shown in the next figure.

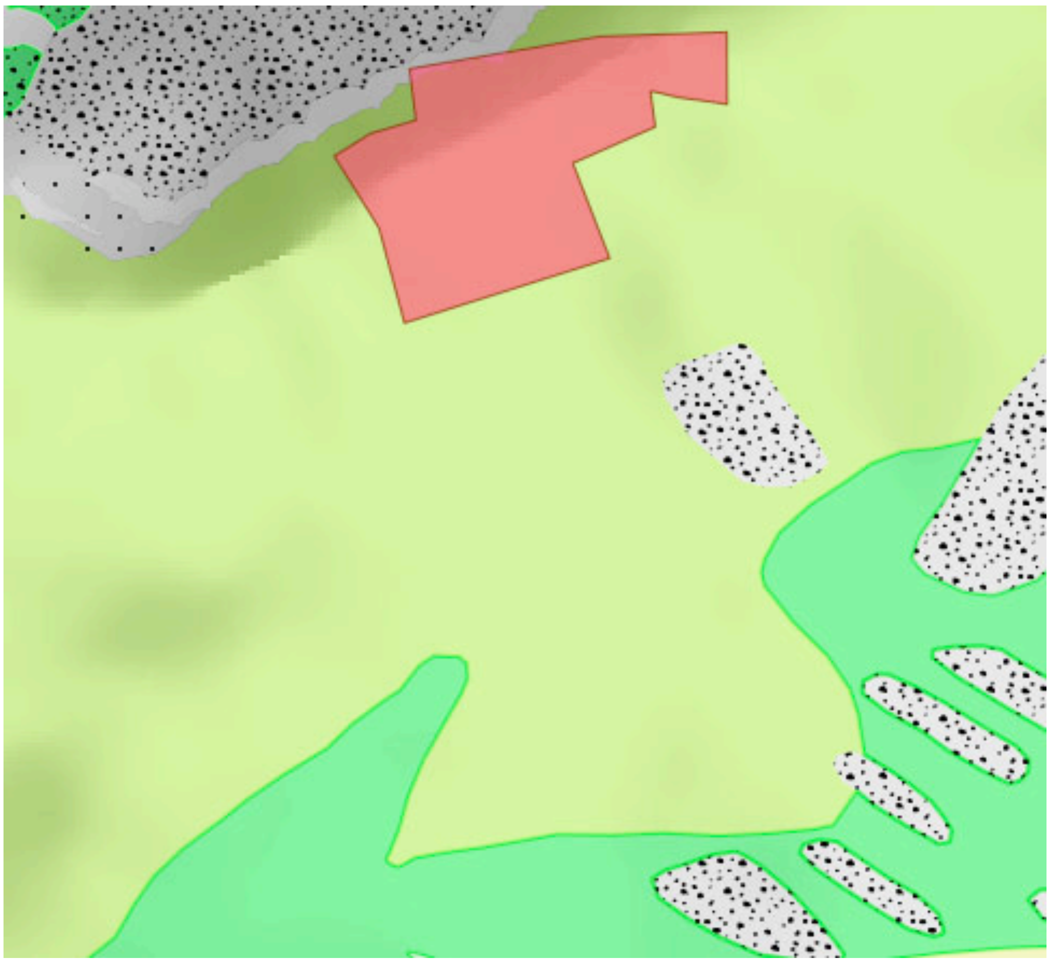


Figure 16. Result of more complex pattern definitions in eliminating pattern repetition

4.4.4 Symbolization of Linear Features

The Stroke element applies to any line, and not just to the borders of polygon geometries inside a PolygonSymbolizer. As a consequence, the Stroke element is the most important part of the LineSymbolizer, which is used specifically for the styling of linear geometries. The LineSymbolizer defines the symbolization of linear geometry types, having the formal definition as follows:

```
<xsd:element name="LineSymbolizer" type="se:LineSymbolizerType" substitutionGroup="se:Symbolizer"/>
<xsd:complexType name="LineSymbolizerType">
   <xsd:complexContent>
      <xsd:extension base="se:SymbolizerType">
         <xsd:sequence>
            <xsd:element ref="se:Geometry" minOccurs="0"/>
            <xsd:element ref="se:Stroke" minOccurs="0"/>
            <xsd:element ref="se:PerpendicularOffset" minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

Advanced graphical output as supported by SVG (vector-based definition for pattern and gradients) is also included in the extensions for line symbolization. The same syntax and meaning as detailed for the polygon Fill pattern is also valid for the linear Stroke pattern with the small additional explanation that the placement parameter x specifies the placement gap for the repeating pattern tile along the line and the parameter y specifies the placement perpendicular to the line. As a consequence of the introduction of line patterns, the definition of the Stroke element is extended as follows:

```
<xsd:element name="Stroke" type="se:StrokeType"/>
<xsd:complexType name="StrokeType">
   <xsd:sequence>
      <xsd:choice minOccurs="0">
         <xsd:element ref="se:GraphicFill"/>
         <xsd:element ref="se:GraphicStroke"/>
      </xsd:choice>
      <xsd:element ref="se:SvgParameter" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="se:SvgRadialGradient" minOccurs="0"/>
      <xsd:element ref="se:SvgLinearGradient" minOccurs="0"/>
      <xsd:element ref="se:SvgPattern" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

4.4.5 Standard SE Support for Dashes versus Patterns

The OGC SE standard currently defines the use of the following SVG/CSS styling parameters for a stroke: “stroke” (color), “stroke-opacity”, “stroke-width”, “stroke-linejoin”, “stroke-linecap”, “stroke-dasharray”, and “stroke-dashoffset”. The “stroke-linejoin” and “stroke-linecap” SvgParameter elements encode how line strings should be joined between line segments and capped at the two ends of the line string [OGC, 2005b]. The “stroke-dasharray” SvgParameter element encodes a dash pattern as a series of dashes separated by spaces. The “stroke-dashoffset” SvgParameter element specifies the position where the dash sequence starts.

The SE standard support for dashes can be illustrated also with fragment of the map representation for the topographic map of Methana, corresponding to the visualization of supplementary contour lines that were already visible in *Figure 11*. In this example, the supplementary contour lines are symbolized with a dash pattern defined with a Stroke which is drawn 4 pixel long and followed by a gap of 3 pixels as defined below:

```
<LineSymbolizer xmlns="http://www.opengis.net/sld">
   <Stroke xmlns="http://www.opengis.net/sld">
      <SvgParameter xmlns="http://www.opengis.net/sld" name="stroke" >#aa5500</SvgParameter>
      <SvgParameter xmlns="http://www.opengis.net/sld" name="stroke-dasharray" >4 3</SvgParameter>
      <SvgParameter xmlns="http://www.opengis.net/sld" name="stroke-width" >1</SvgParameter>
   </Stroke>
</LineSymbolizer>
```

The SVG line patterns are more generic and expressive for cartographic representations. They also include the symbolization possibilities showed above for the dashed lines. For example, a line pattern 4 pixel long followed by a 3 pixels gap similar to above can be defined in a slightly more compact form using a pattern:

```
<pattern width="4" height="1" x="3" y="0">
```

```
    <svg>
        <line x1="0" y1="0.5" x2="4" y2="0.5" style="stroke:#aa5500;stroke-width:1"/>
    </svg>
</pattern>
```

4.4.6 The Usefulness of Line Patterns

Although not yet tested in software, the SVG line patterns are expected to provide sufficient flexibility for the more demanding cartographic aspects of line drawing. By using SVG patterns for Stroke, map representations can be defined that resemble texture mapping for lines as presented in [He, 2008].

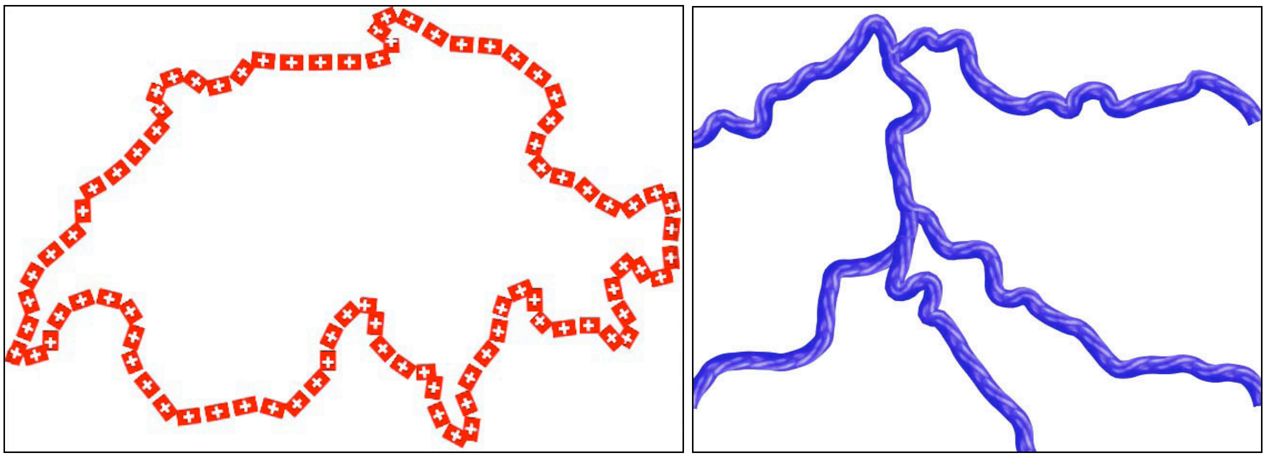


Figure 17. Texture mapping for lines [He, 2008]

For example, the representation for the border of Switzerland can be obtained using a pattern with a gap of 2 and with the flag of Switzerland as SVG content:

```
<pattern width="8" height="6" x="2" y="0">
    <!-- SVG content using http://commons.wikimedia.org/wiki/Image:Flag_of_Switzerland.svg   -->
</pattern>
```

The river network in the second part of the figure could be also expressed with a pattern containing a base blue solid Stroke and a pattern with a negative gap (e.g. x="-3") in order to minimize the artifacts that may appear at the junctions between the line segments.

Moreover, the line patterns can be used to create complex lines, as for example double lines. In the pure OGC SE standard, doubles line can be faked using a workaround. A first FeatureTypeStyle defines the lines to be drawn in black colour with a stroke-width of 5 pixels. A second FeatureTypeStyle defines lines to be drawn in white colour with a stroke-width of 3 pixels. When the two styles are drawn, the white stroke covers the centre of the black stroke, thus creating the visual appearance of a double line. This effect can be observed in the representation of roads for the topographic map of Methana from Figure 18.

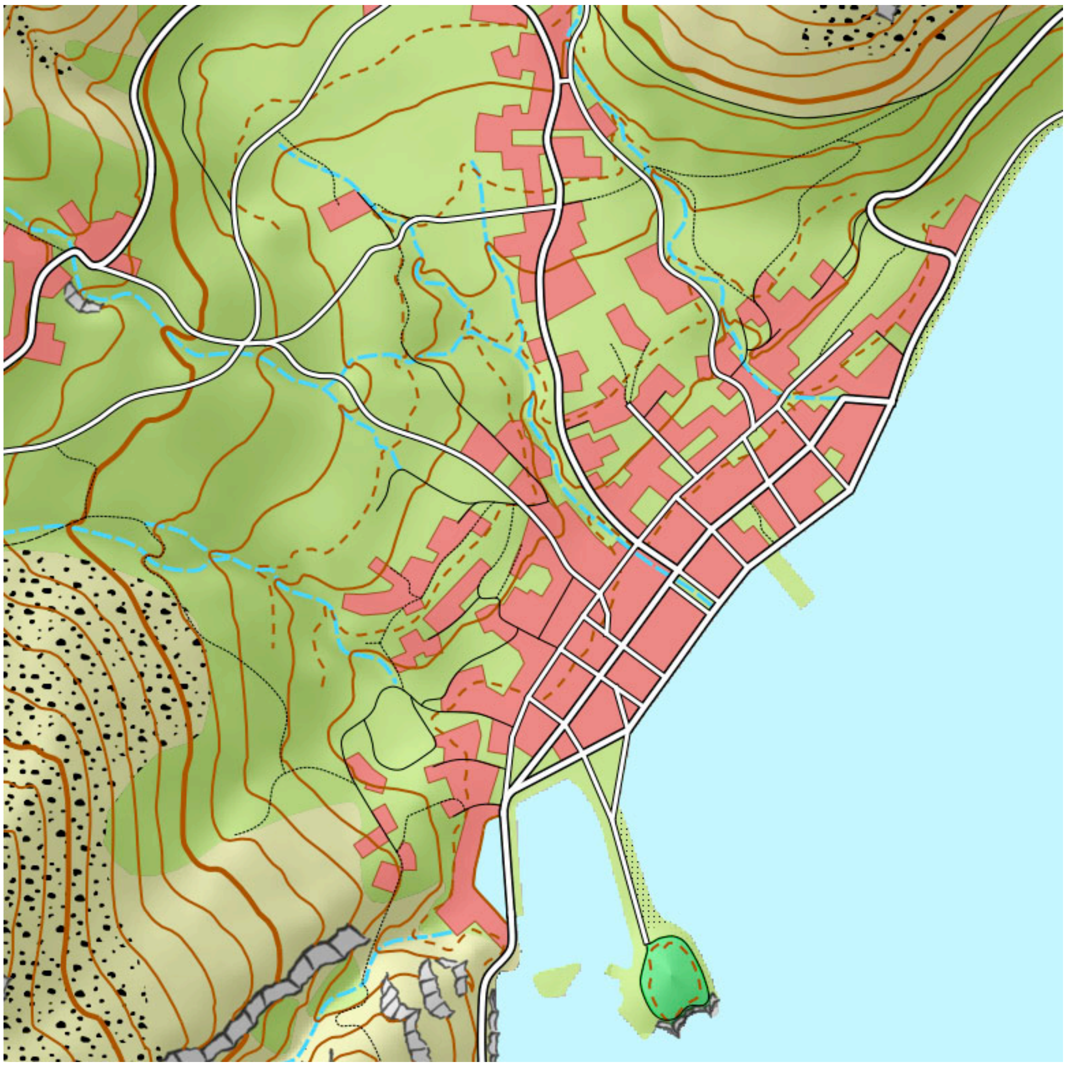


Figure 18. SE "fake" double lines for the symbolization of roads

However, the doubles lines are not transparent in the centre, therefore not true double lines. With the use of a SVG pattern containing two parallel lines, true representations of double lines in the map can also be obtained. Furthermore, the use of several SVG lines patterns contained in FeatureTypeStyle elements and arranged with meaningful longitudinal and perpendicular offsets can truly produce intricate symbolizations for lines.

Finally, it must be noted that the symbolization of lines is encountered for more than linear features/geometries. For example, the Stroke element complements the Fill element and is used for outlining the border of the polygon geometry. For area symbolization it follows conceptually after the Fill element. Therefore, if both Fill and Stroke elements are present in a Polygon Symbolizer, then they are to be represented according to the "painters model" with the Stroke being rendered on top of the fill. Exceptions to this model can be defined in complex map signatures overlaying multiple FeatureTypeStyles.

4.4.7 Symbolization of Points with SVG Symbols

In the SE standard, the symbolization of point geometries is achieved with the PointSymbolizer, defined formally as:

```
<xsd:element name="PointSymbolizer" type="se:PointSymbolizerType" substitutionGroup="se:Symbolizer"/>
<xsd:complexType name="PointSymbolizerType">
  <xsd:complexContent>
    <xsd:extension base="se:SymbolizerType">
      <xsd:sequence>
        <xsd:element ref="se:Geometry" minOccurs="0"/>
        <xsd:element ref="se:Graphic" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

In SE, the Graphic element defines the actual point symbol. The formal definition of the Graphic element in the XML Schema is shown below, with parameters corresponding to names and semantics of the SE standard.

```
<xsd:element name="Graphic" type="se:GraphicType"/>
<xsd:complexType name="GraphicType">
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="se:ExternalGraphic"/>
      <xsd:element ref="se:Mark"/>
    </xsd:choice>
    <xsd:element ref="se:Opacity" minOccurs="0"/>
    <xsd:element ref="se:Size" minOccurs="0"/>
    <xsd:element ref="se:Rotation" minOccurs="0"/>
    <xsd:element ref="se:AnchorPoint" minOccurs="0"/>
    <xsd:element ref="se:Displacement" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

However, the SE defines only basic markers having a WellKnownName element that gives the name of the shape of the mark. Allowed values include “square”, “circle”, “triangle”, “star”, “cross”, and “x”. The WellKnownName element allows to customize the color of the Stroke and the Fill, but the shape of the mark is fixed (as given by the WellKnownName element). The graphical aspect of a “square” mark is similar to the example shown in Figure 19.

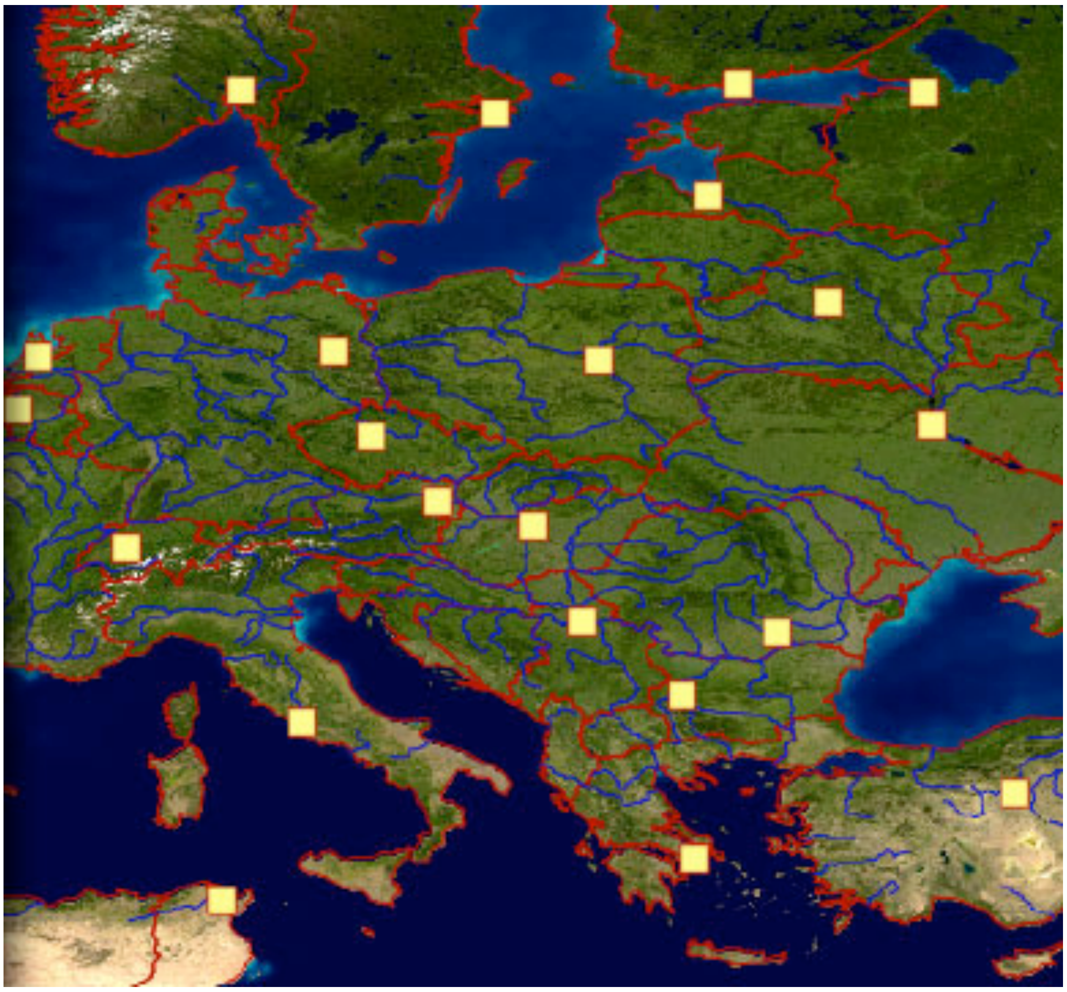


Figure 19. Yellow “square” marks symbolizing European capitals [Iosifescu et al., 2010b]

The alternative to a WellKnownName is an external or in-lined graphic format. In our opinion, the in-line content is not clearly specified and theoretically allows the inclusion of virtually any raster or vector graphics (including SVG). As a result, the software implementations may handle it as “a little picture”, thus limiting its scalability. Moreover, a disadvantage of using external graphic formats is that the map representation is not self-contained. Therefore, we propose to enhance the “Mark” element explicitly with SVG (Scalable Vector Graphics) point symbol definitions for a coherent SVG enhancement of the current SE standard as follows:

```
<xsd:element name="Mark" type="se:MarkType"/>
<xsd:complexType name="MarkType">
  <xsd:sequence>
    <xsd:choice minOccurs="0">
      <xsd:element ref="se:WellKnownName"/>
      <xsd:element ref="se:SvgSymbol"/>
      <xsd:sequence>
        <xsd:choice>
          <xsd:element ref="se:OnlineResource"/>
          <xsd:element ref="se:InlineContent"/>
        </xsd:choice>
        <xsd:element ref="se:Format"/>
        <xsd:element ref="se:MarkIndex" minOccurs="0"/>
      </xsd:sequence>
    </xsd:choice>
    <xsd:element ref="se:Fill" minOccurs="0"/>
    <xsd:element ref="se:Stroke" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="WellKnownName" type="xsd:string"/>
<xsd:element name="SVGElement" type="svg:svg"/>
<xsd:element name="MarkIndex" type="xsd:integer"/>
```

Optionally, the exclusive use of SVG symbols can be enforced with the following strict “Mark” definition. This above definition however, would break the backwards compatibility of the map representation for cartography with the SE standard.

```
<xsd:element name="Mark" type="se:MarkType"/>
<xsd:complexType name="MarkType">
  <xsd:sequence>
    <xsd:choice minOccurs="0">
      <xsd:element ref="se:WellKnownName"/>
      <xsd:element ref="se:SvgSymbol"/>
    </xsd:choice>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="SVGElement" type="svg:svg"/>
<xsd:element name="WellKnownName" type="xsd:string"/>
```

The SvgSymbol element can contain any valid SVG code, including gradients and patterns. Simple point signatures in extended SE can be created using SVG paths, and more complex definitions can contain symbols styled with SVG patterns, gradients and filters.

4.4.8 The Usefulness of SVG Point Symbols

The usefulness and flexibility of point symbolisation with SVG symbols is also illustrated with a step from the reconstruction of the map of Methana. In this map, the buildings (polygon, line and points) are some of the final layers. In the case of the buildings polygon layer, multiple rules using the propertyName BUILDING_T can be used for discriminating between various types of the buildings (e.g. building, cemetery, ruin, sports place, water tank). Then each type is symbolized in a different manner with a PolygonSymbolizer as appropriate for area features. The buildings line layer is also symbolized with the use of LineSymbolizer, which gives us a starting point before demonstrating the use of SVG symbols as shown in Figure 20.

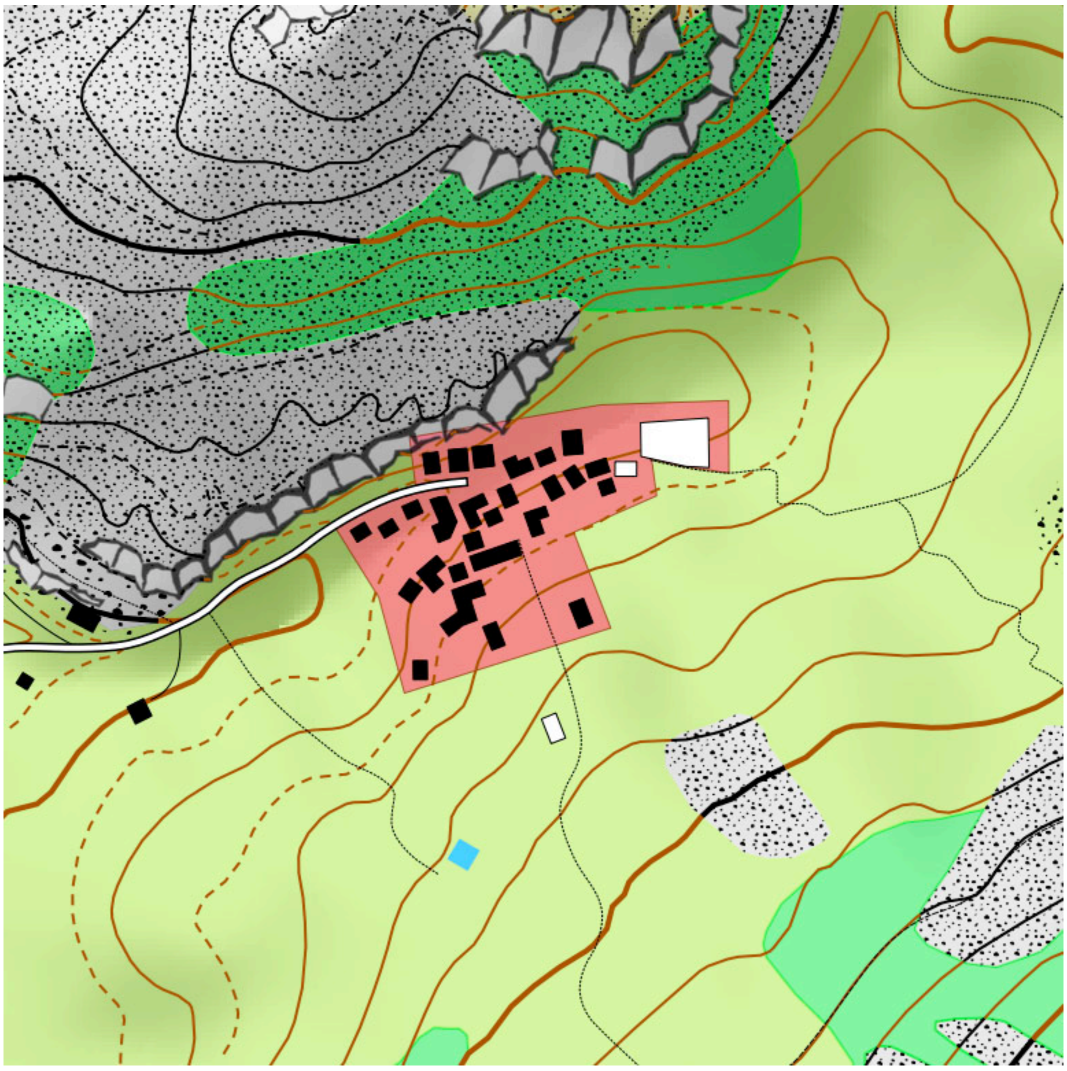


Figure 20. Starting point before the use of SVG point symbols

Starting from the above situation, we can define a map representation using SVG symbols to draw the conventional symbols for building, which have their positions defined in a point layer. In the Methana example, an SVG symbol allows us to draw the conventional point symbol for chapel on the white building, thus resulting in the following map representation fragment:

```
<PointSymbolizer xmlns="http://www.opengis.net/sld">
      <Graphic>
         <Mark>
            <SvgSymbol>
               <!-- SVG symbol for chapel   -->
            </SvgSymbol>
         </Mark>
         <Size>20</Size>
         <Rotation>
            <PropertyName xmlns="http://www.opengis.net/ogc">ET_ANGLE</PropertyName>
         </Rotation>
      </Graphic>
   </PointSymbolizer>
```

In addition, the placement of the symbol was enhanced with a reference to an appropriate rotation angle that was specified in order to obtain the desired alignment of the conventional symbol with the building footprint, as shown in Figure 21.

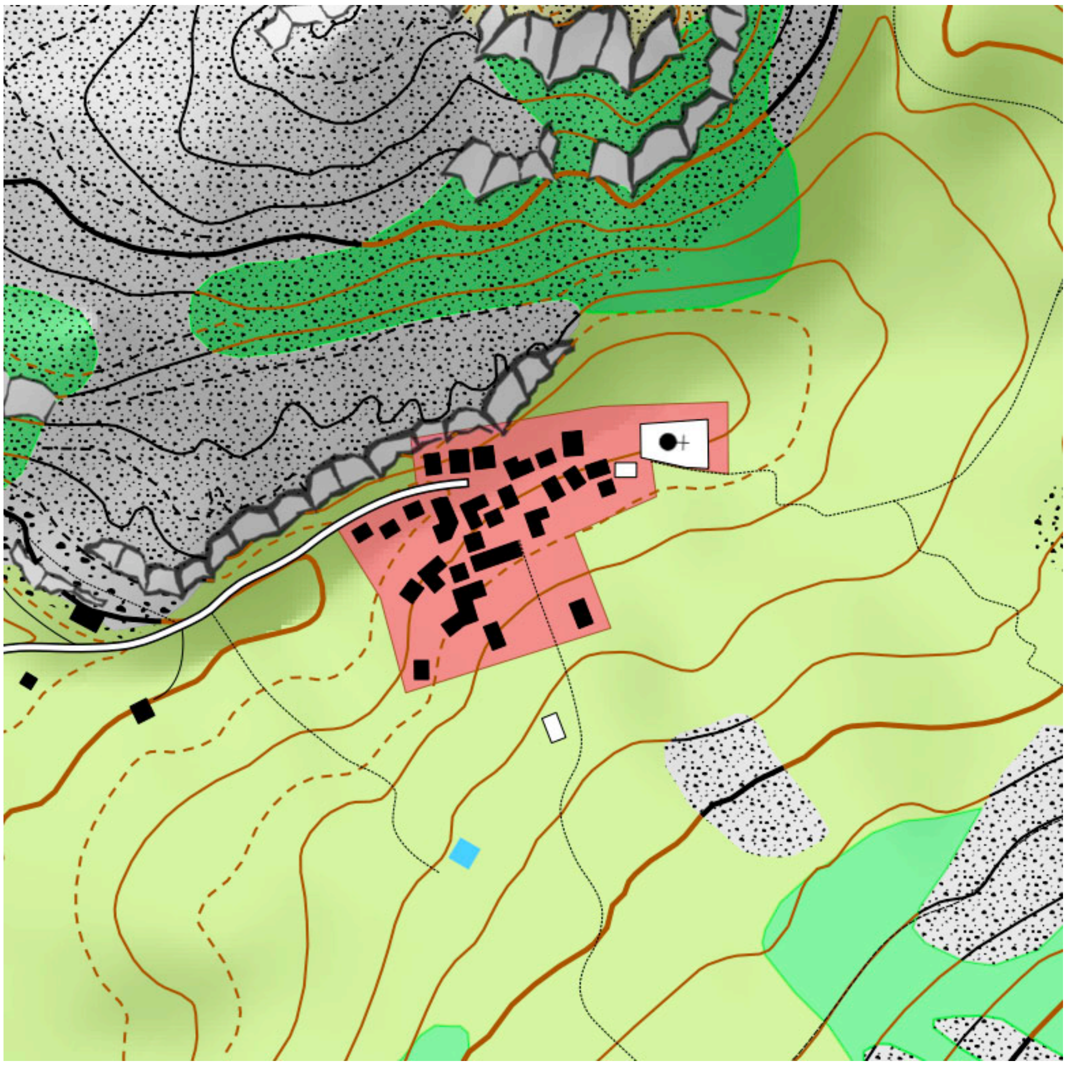


Figure 21. Conventional point symbol for chapel created with an SVG Symbol and rotated for alignment with the building footprint

As shown above, the expression in a map representation of the conventional symbol for chapel is clear when expressed with an SVG symbol.

4.4.9 Map Representation for Labels

The map representation for cartography integrates the SE standard for labeling without extensions. Labeling is achieved with the SE TextSymbolizer, which describes the representation of labels on the map. The Text Symbolizer is defined formally as:

```
<xsd:element name="TextSymbolizer" type="se:TextSymbolizerType" substitutionGroup="se:Symbolizer"/>
<xsd:complexType name="TextSymbolizerType">
   <xsd:complexContent>
      <xsd:extension base="se:SymbolizerType">
         <xsd:sequence>
            <xsd:element ref="se:Geometry" minOccurs="0"/>
            <xsd:element ref="se:Label" minOccurs="0"/>
            <xsd:element ref="se:Font" minOccurs="0"/>
            <xsd:element ref="se:LabelPlacement" minOccurs="0"/>
            <xsd:element ref="se:Halo" minOccurs="0"/>
            <xsd:element ref="se:Fill" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

In the SE standard, the Label element references the text-label content. The Font element identifies a font of a certain family, style, and size. The font-family element specifies the name of the font. Allowed values for the font-style are normal, italic, underlined, oblique, and allowed values for font-weight are normal and bold. Font-size SvgParameter element specifies the size to use for the font in pixels or millimeters. The LabelPlacement element is used to position a label relative to a point, line string or polygon. A Halo is a type of Fill that is applied to the backgrounds of font glyphs for improving the readability of text labels. Finally, the label representation is encapsulated in TextSymbolizer.

An example from the reconstruction of the topographic map of Methana, presented in the following fragment for map representation which allows several points of interests (such as the chapel) to be annotated with their name using an italic font style:

```
<TextSymbolizer>
  <Geometry>
    <ogc:PropertyName>LABELPOS</ogc:PropertyName>
  </Geometry>
  <Label>
    <ogc:PropertyName>Name</ogc:PropertyName>
  </Label>
  <Font>
    <SvgParameter name="font-family">Arial</SvgParameter>
    <SvgParameter name="font-family">Sans-Serif</SvgParameter>
    <SvgParameter name="font-style">italic</SvgParameter>
    <SvgParameter name="font-size">26</SvgParameter>
  </Font>
  <Halo>
    <Radius>1</Radius>
    <Fill>
      <SvgParameter name="fill">#ffffff</SvgParameter>
    </Fill>
  </Halo>
  <Fill>
    <SvgParameter name="fill">#000000</SvgParameter>
  </Fill>
</TextSymbolizer>
```

The result of the map representation fragment above translates in a basic labeling of features in the map. Moreover, the effects of the various TextSymbolizer options are visible in the next figure:

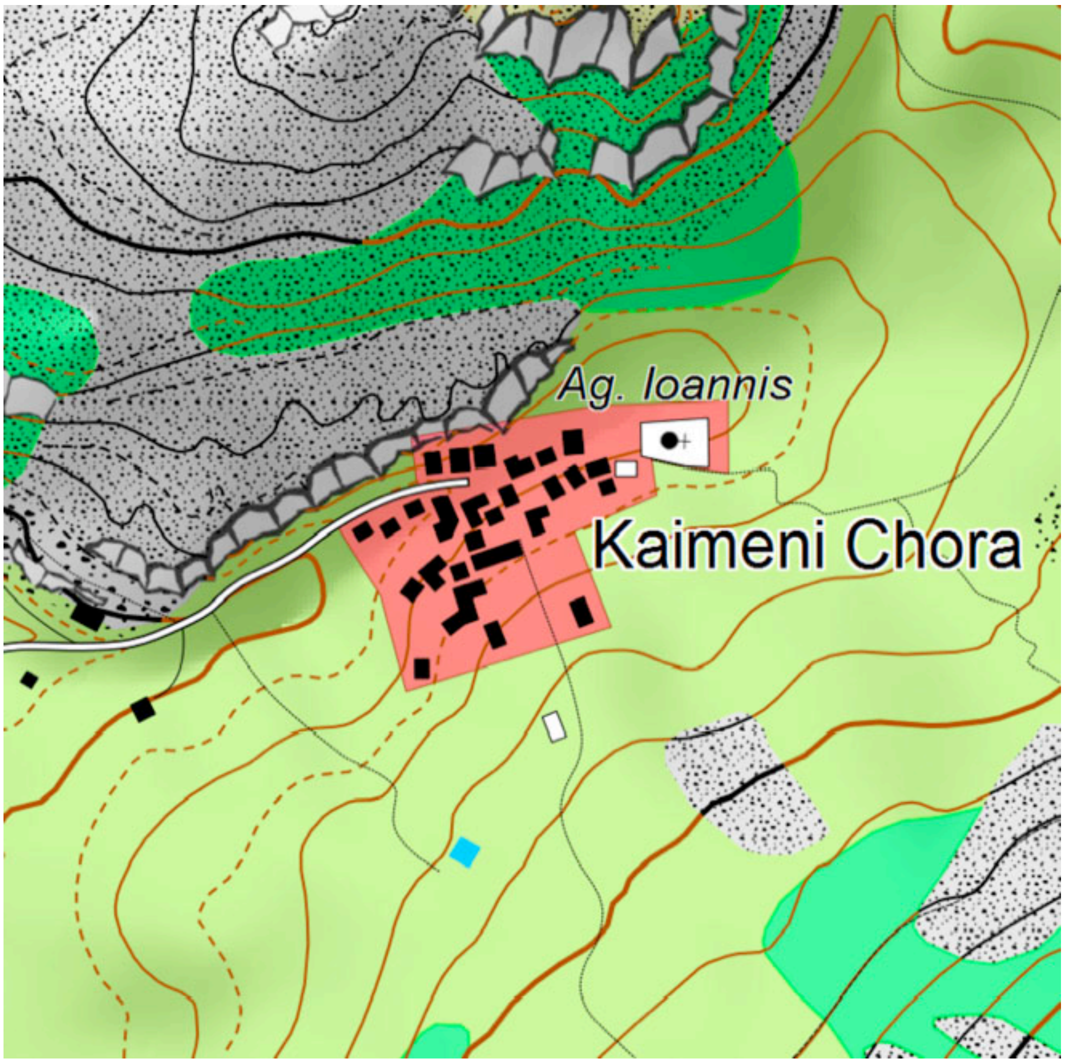


Figure 22. Example illustration of various options for the representation of labels

4.5 Map Representation for Thematic Data

4.5.1 Overview on the Symbolization Possibilities for Thematic Data

In general, the fundamental possibilities for representing thematic data in a map representation are proportional symbol maps and choropleths. Although diagram symbolization of geographic features is an effective way of representing thematic data in a spatial context, the current OGC SE standard does not support representations of multiple data values using diagrams or proportional symbols at all. As a consequence, the map representation for cartography contains extensions specifically designed to fill this gap. In addition, the definition of choropleth maps through a map representation is optimized for an overall improvement of thematic representations.

4.5.2 Map Representation for Choropleth Maps

Considering the creation of choropleth maps, the representation expressed in pure SE has to be specified in a very lengthy description of the individual classes and their thresholds. There are no classification options for choropleths or predefined diagram types, which make the creation of thematic maps straightforward in current desktop GIS software [Iosifescu, 2007c].

However, classification methods would allow interactive Web mapping clients to easily serve also as exploratory data visualization (EDV) backend tools. Their major advantage would be their conciseness in defining a choropleth representation. It would allow users to get a first impression of the spatial distribution of the attribute data as illustrated in Figure 23 with just a few lines of map representations, a feature that would enrich the interactions with the maps for EDV.

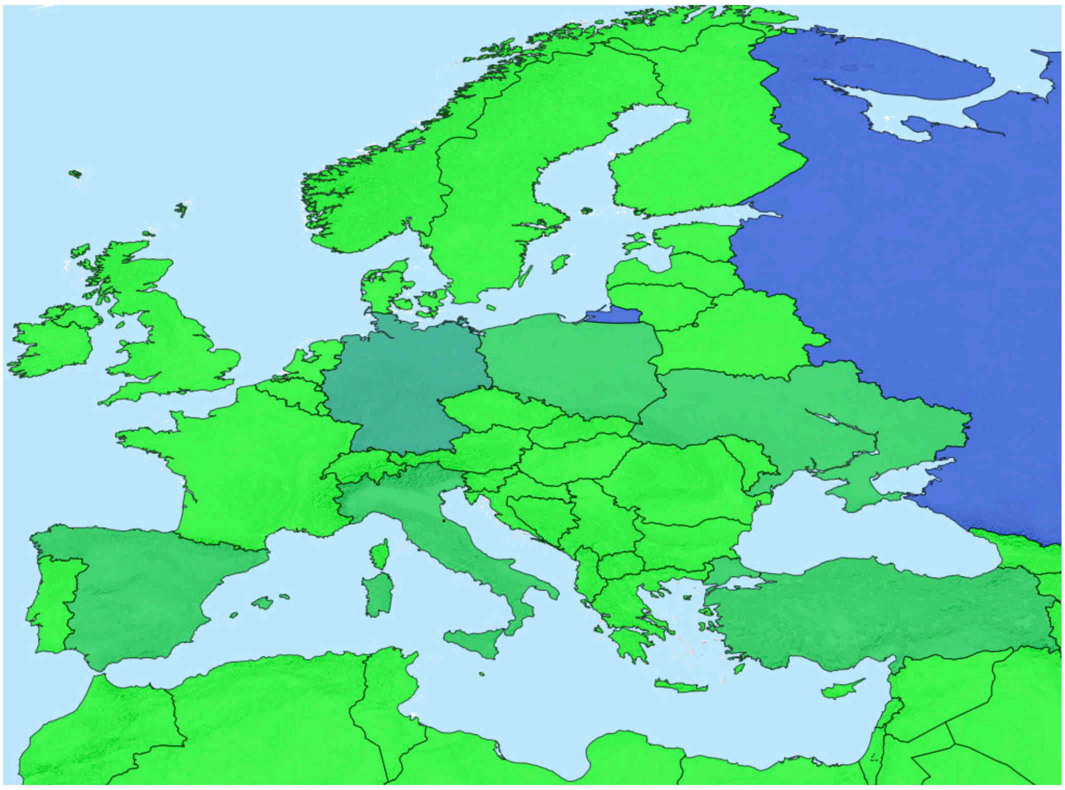


Figure 23. Example of a choropleth map

Therefore, a simple extension of the SE with symbology definitions to support thematic choropleth in a clear and interoperable manner for cartography was performed. The Classification element for a concise choropleth map definition can be added to the SE "Rule" element. A Method element contains the name of the classification method to be applied. Allowed values include "EqualInterval", "NaturalBreaks", "Quantile", "StandardDeviation" with the same semantic and definition known from desktop GIS software. The default Method is "EqualInterval". A more detailed specification of the choropleth maps can be found in [Iosifescu, 2007c], which contains among other thematic extensions, the formal specification of the new choropleth definition for SE submitted to OGC as an SE Implementation Specification Change Request (CR).

4.5.3 Generic Definition for Proportional Symbol Maps and Diagrams

In a basic form, the SvgSymbol element introduced for the symbolization of point features can be successfully used also for creating any kind of proportional symbol maps. For example, it is given the following basic map representation for a PointSymbolizer:

```
<PointSymbolizer xmlns="http://www.opengis.net/sld">
       <Graphic>
          <Mark>
             <SvgSymbol>
                <!-- SVG symbol for a blue sphere with gradient  -->
             </SvgSymbol>
          </Mark>
          <Size>
            <PropertyName xmlns="http://www.opengis.net/ogc">tot_pop</PropertyName>
          </Size>
       </Graphic>
  </PointSymbolizer>
```

The fragment of the map representation presented above uses an SVG Symbol for a blue sphere, which size is defined to be proportional to an attribute in the point data layer. Assuming we are going to add this symbolization on top of the choropleth map defined previously, the resulting visualization is a map with a proportional symbol layer shown in Figure 24.

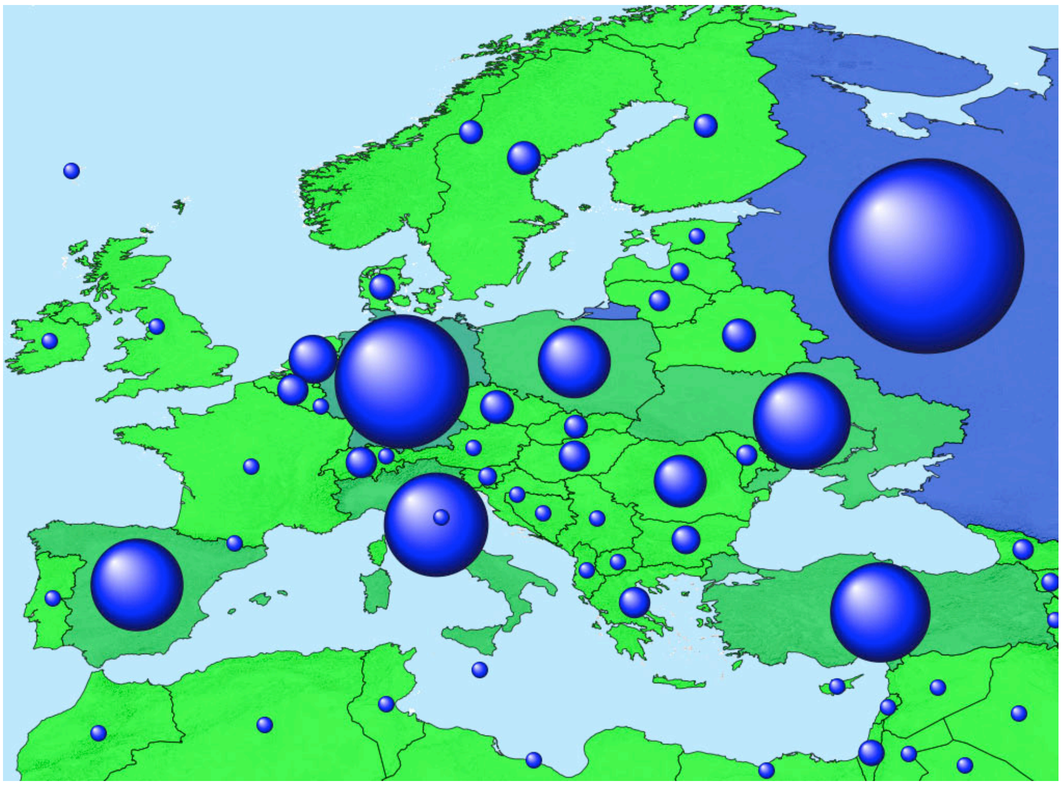


Figure 24. Example of a proportional symbol map using SVG symbols

Moreover, several FeatureTypeStyle elements containing displaced SVG point symbols would allow the contraction of any point signature, including various types of diagrams regardless of their complexity. This is however highly unpractical. Albeit possible, the use of the SvgSymbol for these diagram maps would require extensive efforts for defining all the parts of the diagram signatures in order to compare to the diagram signatures defined by [Schnabel, 2007].

4.5.4 The Diagram Symbolizer

The Diagram Symbolizer is the most important thematic extension. The core of this symbolizer is the Diagram element, a graphic symbol suitable for thematic mapping such as proportional symbols and diagrams maps. The Diagram element is defined as follows:

```
<xsd:element name="Diagram" type="se:DiagramType"/>
<xsd:complexType name="DiagramType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:sequence>
        <xsd:element ref="se:WellKnownName"/>
        <xsd:element ref="se:Subtype" minOccurs="0"/>
        <xs:element ref="se:Category" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="se:3D" minOccurs="0"/>
      <xsd:sequence>
      <xsd:element ref="se:SvgSymbol"/>
    </xsd:choice>
    <xsd:element ref="se:Size" minOccurs="0"/>
    <xsd:element ref="se:Scale" minOccurs="0"/>
    <xsd:element ref="se:Opacity" minOccurs="0"/>
    <xsd:element ref="se:Rotation" minOccurs="0"/>
    <xsd:element ref="se:AnchorPoint" minOccurs="0"/>
```

```
      <xsd:element ref="se:AnchorLine" minOccurs="0"/>
      <xsd:element ref="se:Displacement" minOccurs="0"/>
      <xsd:element ref="se:Halo" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

In the above introduction of the DiagramSymbolizer, it can be noticed that there can be defined different diagram types, the WellKnownName element specifying the type of the diagram. Allowed values include at least “Pie”, “Bar”, “Line”, “Area”, “Ring”, and “Polar”, though map servers may have additional (specialized) ones. The default WellKnownName is “Pie”. The Subtype element gives the subtype of the diagram. Allowed values include “Normal”, “Stacked” and “Percent”, though map servers may have additional (specialized) ones. The default Subtype is “Normal”.

The SvgSymbol element contains valid SVG code (including SVG-embedded raster images if desired) in a similar manner as integrated in the PointSymbolizer. This time it defines a symbol to be used for the representation of thematic data values. The SvgSymbol element is to be used only if no diagrams are necessary as in the case of simple proportional symbol maps. This restriction is stated in order to prevent the complexity problems discussed in the previous subchapter for creating diagrams with the use of the SVG Symbol from the PointSymbolizer. An overview of several thematic representations that are possible to be defined using a DiagramSymbolizer is presented in Figure 25 [Ortner, 2011].

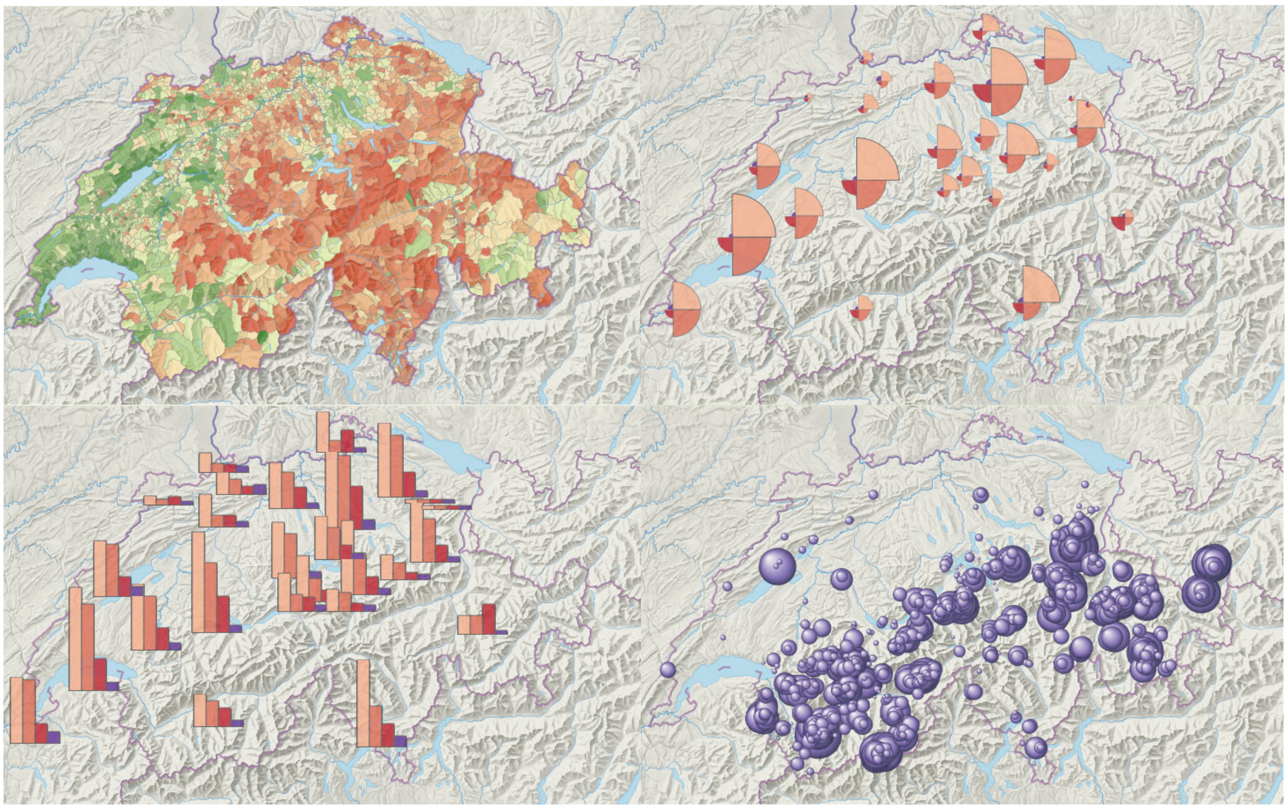


Figure 25. Overview of several thematic representations defined using a DiagramSymbolizer [after Ortner, 2011]

The Category element of the Diagram defines one of multiple values from an ogc:PropertyName to be represented in the diagram. The Gap element is an optional gap distance given to the Category. It is to be used by the various diagram types to define a space between the various diagram categories. For example, for Bar diagrams it represents a gap between bars and for Pie diagrams it represents how much a category should be offset.

The Size element gives the absolute size of the graphic in a specified unit of measurement (uom). It contains the ogc:PropertyNames it uses to map values to absolute sizes or a fixed absolute size can be given. If more than one ogc:PropertyName element is encountered than the sum of all the numerical ogc:PropertyNames is considered. This feature allows to express the size of the diagram proportional to the total sum of the categories but also to define a simple proportional symbol. The Categorize and Interpolate elements have the same definition as in the latest SE specifications. The use of Categorize element is more suitable to be used for the presentation of qualitative data because it requires clearly defined data categories. The Interpolate element is more suitable for the presentation of quantitative data, since graphic properties such as color will be interpolated for all values in between the given threshold data values.

An additional effect of using these elements is the implicit definition of the maximum and minimum symbol sizes. The Scale element allows the scaling of the defined absolute size of the diagrams or symbols to the relative map scale, with the scale denominators MinScaleSizeMultiplication and MaxScaleSizeMultiplication representing multiplication factors that compensate for map scale changes.

Finally, we have to consider the placement of diagrams on the map. A DiagramSymbolizer is used to draw a diagram at a point. Point geometries attached to the thematic data are assumed to exist in order to define the optimal placement of the diagrams. The DiagramSymbolizer will use the centroid of the line and polygon geometries when such geometries are available instead of point geometries. A different anchor point can also be specified to avoid conflicts between diagram/labels. An AnchorLine element specifically instructs the map server to de-conflict diagram/labels by displacement as best as possible and to draw additional anchor lines to clearly associate the diagram/symbol with its logical geometry point on the map.

In addition to diagram maps, it should be also possible to construct proportional symbol maps and dot density maps using the DiagramSymbolizer. However, complex diagram types cannot be expressed in the map representation (except the ones that were defined above). As mentioned in the beginning, based on the SVG symbol definition also more complex diagrams can, in theory, be constructed by following descriptions of construction principles similar to [Schnabel, 2006]. Even if such descriptions are possible to be created in combination with additional software tools supporting the creation of such descriptions (e.g. a symbol brewer), the results would be too complex to be easily readable by cartographers, which in consequence would erode the use of map descriptions for thematic mapping.

In conclusion, the DiagramSymbolizer effectively extends SE with a thematic mapping profile for point symbols and does not impose unnecessary changes in existing SLD/SE standards, thus completing the specification of a map representation for topographic and thematic cartography.

More detailed technical specifications, written in the OGC style, about the symbolization possibilities for thematic data can be found in [Iosifescu, 2007c]. This document is an SE Implementation Specification Change Request and contains a detailed specification of the Diagram Symbolizer and other selected SE enhancements as discussed in this work. Furthermore, a compact and less technical overview on the achievements of a map representation for topographic and thematic cartography can be found in [Iosifescu et al., 2010a] and an evaluation of its applicability for thematic atlases can be found in [Ortner, 2011].

4.6 Map Representation for Raster Data

4.6.1 Overview on the Symbolization of Raster Data

Vector data is of main interest in cartography since it requires processing and symbolization through the manipulation of the graphic variables. Therefore, this work focuses on real-time cartographic representation of vector data as the main data source for a map, and as consequence vector data is the main target for map representation. However, the representation of raster data is also important for certain Web cartographic products.

In most cases, pixel maps, shaded reliefs and aerial imagery are usually displayed as they are, thus no additional symbolization is required through a map representation (although specialized GIS or remote sensing software may have been used for producing these rasters). But when the cells of original source raster contain information instead of RGB pixels, it may be useful to have methods for symbolizing the raster values. The most obvious example is when the raster data in question is a Digital Elevation Model (DEM) that can be used to produce shaded reliefs, hypsometric tints, contour lines, profiles, slope and aspect, exposition, relief-dependent interpolation etc.

4.6.2 Symbolization of Raster Data with the RasterSymbolizer

The symbolization of raster data sources can be performed with the extended RasterSymbolizer, which is defined formally in XML Schema as follows:

```
<xsd:element name="RasterSymbolizer" type="se:RasterSymbolizerType" substitutionGroup="se:Symbolizer"/>
<xsd:complexType name="RasterSymbolizerType">
   <xsd:complexContent>
      <xsd:extension base="se:SymbolizerType">
         <xsd:sequence>
            <xsd:element ref="se:Geometry" minOccurs="0"/>
            <xsd:element ref="se:Opacity" minOccurs="0"/>
            <xsd:element ref="se:ChannelSelection" minOccurs="0"/>
            <xsd:element ref="se:OverlapBehavior" minOccurs="0"/>
            <xsd:element ref="se:ColorMap" minOccurs="0"/>
            <xsd:element ref="se:ContrastEnhancement" minOccurs="0"/>
            <xsd:element ref="se:ShadedRelief" minOccurs="0"/>
            <xsd:element ref="se:Isolines" minOccurs="0"/>
            <xsd:element ref="se:Slope" minOccurs="0"/>
            <xsd:element ref="se:Aspect" minOccurs="0"/>
            <xsd:element ref="se:ImageOutline" minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The ColorMap element defines the mapping of values to colors. There are two options for the assignment of colors to pixel values: interpolated and categorized. The Categorize SE option allows raster symbolization with fixed colors according to the classification of values while the Interpolate SE function describes that the colors should be interpolated between values, thus achieving gradual color change from one threshold value to another (gradients).

4.6.3 Isolines, Aspect and Slope

In the definition of the Raster Symbolizer new elements like Isolines, Aspect and Slope are also defined. The Isolines element, which can be used to create contours directly from a DEM, has the following formal definition:

```
<xsd:element name="Isolines" type="se:IsolinesType"/>
<xsd:complexType name="IsolinesType">
   <xsd:sequence>
      <xsd:element ref="se:MaxValue" minOccurs="0"/>
      <xsd:element ref="se:MinValue" minOccurs="0"/>
      <xsd:element ref="se:Equidistance" minOccurs="0"/>
      <xsd:element ref="se:Offset" minOccurs="0"/>
       <xsd:element ref="se:LineSymbolizer" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="MinValue" type="xsd:double"/>
<xsd:element name="MaxValue" type="xsd:double"/>
<xsd:element name="Equidistance" type="xsd:double"/>
<xsd:element name="Offset" type="xsd:double"/>
```

The MinValue and MaxValue specify the values between which the contours are represented. Equidistance specifies the difference between the values (interval) on which the contours will be drawn and the offset allows the user to specify an offset so that additional contours can be drawn also between the rounded values specified by the equidistance, i.e. contours are drawn at a level defined by equidistance plus offset. As the resulting isolines are line features, a LineSymbolizer is controlling the symbolization instead of a ColorMap element.

The Slope and Aspect elements, which can be used to compute the slope and respectively the aspect of a DEM, have the following formal definitions:

```
<xsd:element name="Slope" type="se:SlopeType"/>
<xsd:complexType name="SlopeType">
   <xsd:sequence>
      <xsd:element ref="se:OutputUom" minOccurs="0"/>
       <xsd:element ref="se:ReliefFactor" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="Aspect" type="se:AspectType"/>
<xsd:complexType name="AspectType">
   <xsd:sequence>
      <xsd:element ref="se:OutputUom" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name=" OutputUom " type="xsd:string"/>
```

For both slope and aspect computations, the resulting raster has the default output values expressed in degrees. Radians can be also supported as an alternative angle unit of measurement if specified inside the OutputUom element. The exaggeration factors of the original raster values are expressed using the ReliefFactor element. As an additional note, this element was used because it is already defined in the current SE Specifications, although in the context of the Slope and Aspect the name ReliefFactor is not appropriate. The change of the ReliefFactor element to a more generic name (e.g. ExaggerationFactor) could be considered, if this decision would not collide with existing standards.

4.6.4 Relief Shading

The map representation for cartography should also allow for producing a shaded relief on demand that can be used as the base layer in a map starting from a DEM (available as a raster data source). The ShadedRelief element is defined formally in XML Schema as follows:

```
<xsd:element name="ShadedRelief" type="se:ShadedReliefType"/>
<xsd:complexType name="ShadedReliefType">
   <xsd:sequence>
      <xsd:element ref="se:BrightnessOnly" minOccurs="0"/>
      <xsd:element ref="se:Azimuth" minOccurs="0"/>
      <xsd:element ref="se:Altitude" minOccurs="0"/>
      <xsd:element ref="se:ReliefFactor" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="BrightnessOnly" type="xsd:boolean"/>
<xsd:element name="ReliefFactor" type="xsd:double"/>
<xsd:element name="Azimuth" type="xsd:double"/>
<xsd:element name="Altitude" type="xsd:double"/>
```

The definition of the Shaded Relief is enhanced with parameters like Azimuth and Altitude, which better specify the horizontal and the vertical angle of the light source used in shading algorithms. The ReliefFactor maintains the role of specifying the (vertical) exaggeration value to use. Furthermore, the geometry element that is present in the Raster Symbolizer (either as GML or computed with the aforementioned spatial operators) can be used to specify detailed areas where the parameters take effect. In case of several rules containing relief shading of the same DEM but for different areas (as specified by the geometry), the map server default behavior should be to ensure a smooth transition between the shaded areas.

4.6.5 Example of Map Representation for Relief Shading

Although the enhanced RasterSymbolizer is not implemented in software, we can provide examples that document the specification of a shaded relief in a map representation:

```
<RasterSymbolizer>
   <Opacity>1.0</Opacity>
   <ColorMap>
      <Interpolate>
         <LookupValue>Rasterdata</LookupValue>
         <InterpolationPoint>
            <Data>0</Data>
            <Value>#000000</Value>
            </InterpolationPoint>
            <InterpolationPoint>
                <Data>255</Data>
                <Value>#ffffff</Value>
            </InterpolationPoint>
        </Interpolate>
    </ColorMap>
    <ShadedRelief>
        <Azimuth>315</Azimuth>
        <Altitude>45</Altitude>
        <ReliefFactor>1</ReliefFactor>
    </ShadedRelief>
</RasterSymbolizer>
```

The cartographic rule shown above instructs a cartographic Web service to create a shaded relief with standard illumination settings (azimuth 315 degrees, altitude 45 degrees and without vertical exaggeration).

In addition, a spatial OGC Filter may contains an intersection with an additional geometry in polygon format (e.g. coastlines) can be used to “clip” the relief shading with additional layers in the map representation. A possible result of this cartographic rule is visible in the figure below:

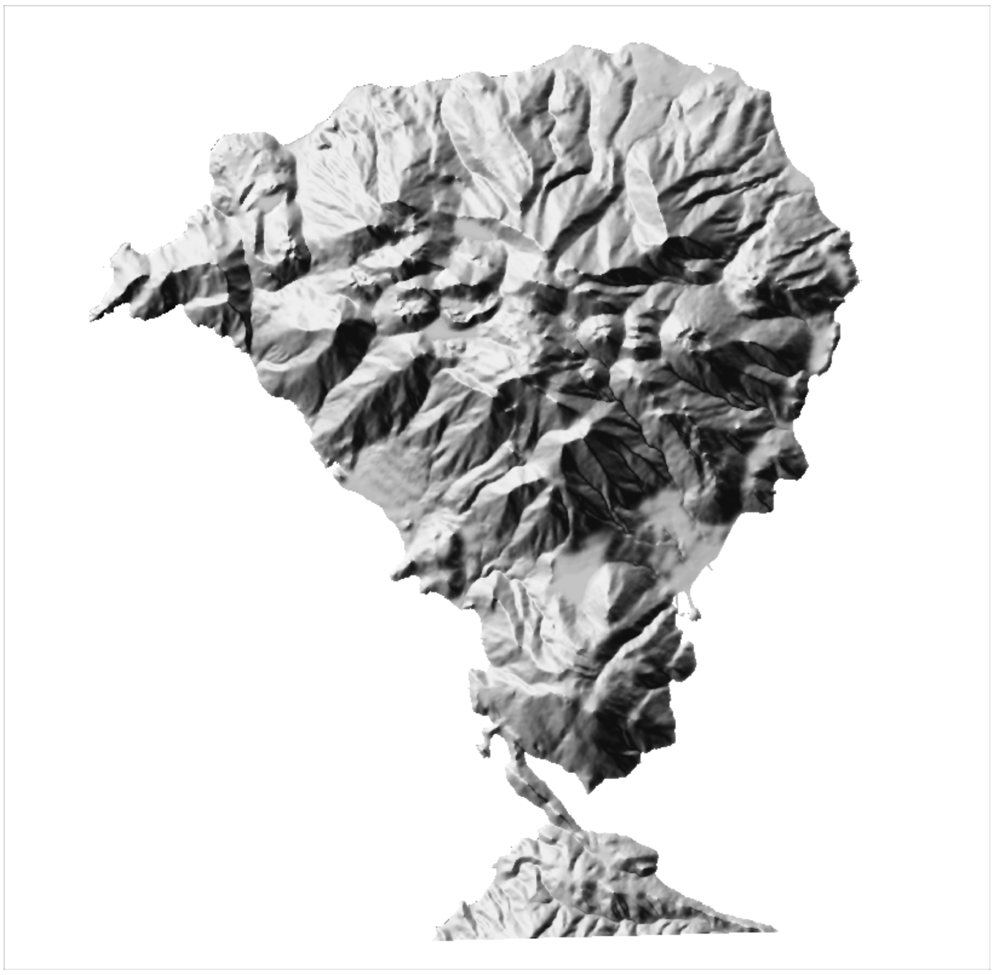


Figure 26. Result of a map representation for relief shading for the topographic map of Methana starting from the DTM and masked by the coastlines

4.6.6 Advanced Relief Shading

By defining multiple layers of symbolization, combined with different opacity levels inside a User Style, it is possible to define advanced symbolizations for shaded reliefs. For example, it is possible to express the enhancement of a shaded relief in contrast with the method defined by Patterson [Patterson et al., 2010] and illustrated in Figure 27.

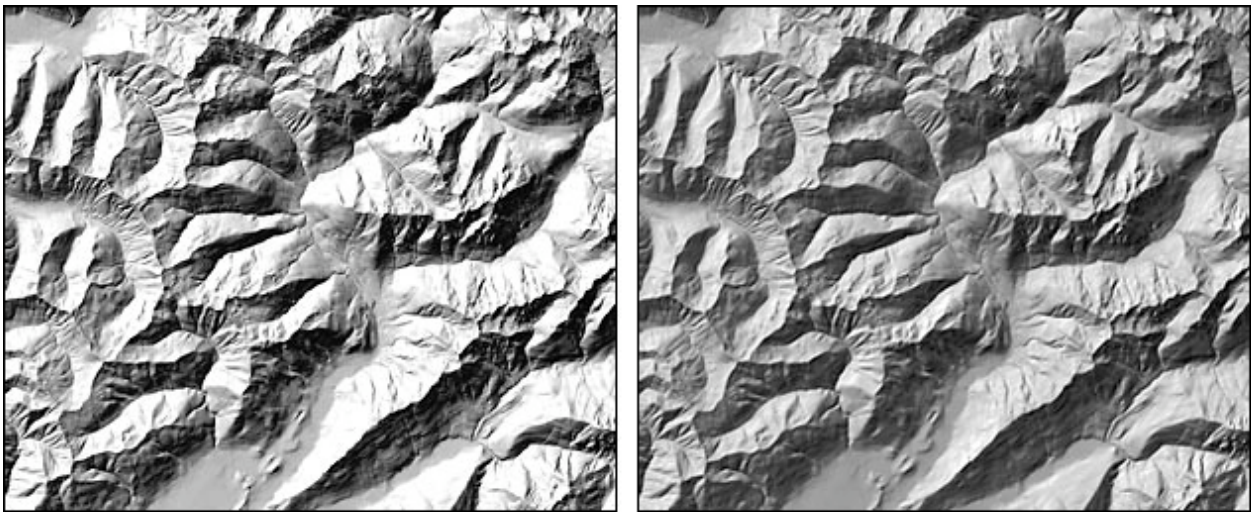


Figure 27. Possible results of cartographic rules for unenhanced (left) and value-enhanced (right) shaded relief [Patterson et al., 2010].

A suggested map representation for the method of relief enhancement [Patterson et al., 2010] is presented in the following:

```
<UserStyle xmlns="http://www.opengis.net/sld">
  <Name xmlns="http://www.opengis.net/sld">ValueEnhanced</Name>
  <CoverageStyle>
    <Rule>
      <Name>Base Relief</Name>
      <Description><Title>Basic relief shading with default illumination parameters</Title></Description>
      <RasterSymbolizer>
        <Opacity>1.0</Opacity>
        <ColorMap>
          <Interpolate>
            <LookupValue>Rasterdata</LookupValue>
            <InterpolationPoint>
              <Data>0</Data>
              <Value>#000000</Value>
            </InterpolationPoint>
            <InterpolationPoint>
              <Data>255</Data>
              <Value>#ffffff</Value>
            </InterpolationPoint>
          </Interpolate>
        </ColorMap>
        <ShadedRelief>
          <Azimuth>315</Azimuth>
          <Altitude>45</Altitude>
          <ReliefFactor>1</ReliefFactor>
        </ShadedRelief>
      </RasterSymbolizer>
    </Rule>
    <Rule>
      <Name>ReliefEnhancement</Name>
      <Description><Title>Relief shading with reduced vertical exaggeration</Title></Description>
      <RasterSymbolizer>
        <Opacity>0.5</Opacity>
        <ColorMap>
          <Interpolate>
            <LookupValue>Rasterdata</LookupValue>
            <InterpolationPoint>
              <Data>0</Data>
              <Value>#000000</Value>
            </InterpolationPoint>
            <InterpolationPoint>
              <Data>255</Data>
              <Value>#ffffff</Value>
            </InterpolationPoint>
          </Interpolate>
        </ColorMap>
        <ShadedRelief>
          <Azimuth>315</Azimuth>
          <Altitude>45</Altitude>
          <ReliefFactor>0.25</ReliefFactor>
        </ShadedRelief>
      </RasterSymbolizer>
    </Rule>
  </CoverageStyle>
</UserStyle>
```

The above map representation defines two shaded relief representations with different vertical exaggerations of the height and then averaged through transparency. When the implementation of software for map representations will become widely available, cartographers may also experiment producing hypsometric tints in a similar manner as for relief enhancement by combining a categorized ColorMap and the ShadedRelief. However, it must be mentioned that much attention should be paid when working with raster data. The resolution of the original DEM has a direct influence on the scale to which a certain shaded relief is represented, as at large scales the structure of the underlying raster data may become visible (it is assumed that the shaded relief implementation does not make the relief blurred or fuzzy at large scales). Therefore the use of a RasterSymbolizer in the map representation should be always used in conjunction with a matching condition for a certain scale range.

4.6 The Map Representation as the Foundation for Cartographic Web Services

The map representation is the core enabler for cartographic web services. In this chapter we have proved that an enhanced SE syntax has the syntactic expressivity required to describe in detail the map-making process based on GIS vector and raster data. The focus lies on vector data as the main map input, however the syntax for expressing raster symbolization was also handled since raster data may be used for the base layers of the map.

The definition of the detailed syntax for expressing the map representation enables us to proceed to the next step in our research, namely the specification of cartographic web services.

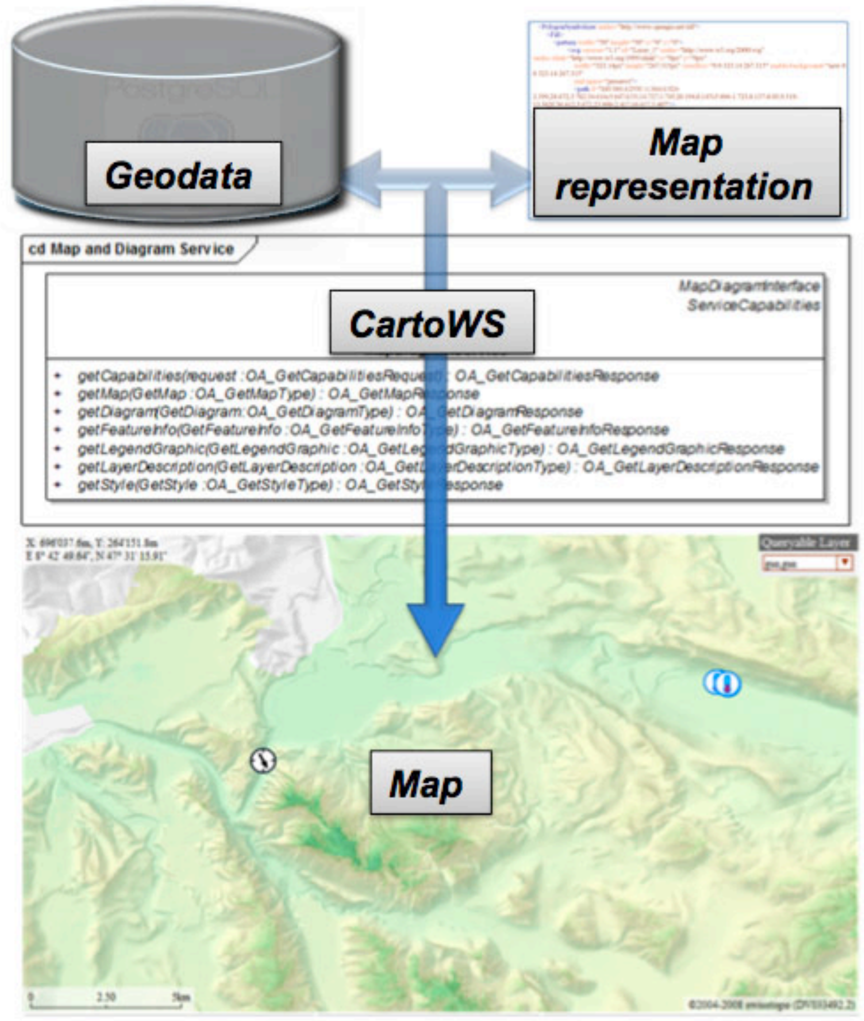
5 Cartographic Web Services

This chapter introduces and defines the cartographic Web services with the notion of service-driven cartography. Then the Map and Diagram Service Interface (MDSI) specifies the generic concept of CartoWS. With the use of basic computer science knowledge, the interface contains formal specifications of the operations of CartoWS. Moreover, it is explained how cartographic web services contribute to the development of cartographic applications.

5.1 The Concept of Cartographic Web Services

5.1.1 Service-driven Cartography

Service-driven cartography can be defined as the usage of services in cartographic application for the core task of map presentation. As shown in *Figure 28*, on the first level we have the geodata and the map representation as the necessary ingredients. The final result is a map that will be displayed in a cartographic Web application.



```
+ getCapabilities(request :OA_GetCapabilitiesRequest) : OA_GetCapabilitiesResponse
+ getMap(GetMap :OA_GetMapType) : OA_GetMapResponse
+ getDiagram(GetDiagram :OA_GetDiagramType) : OA_GetDiagramResponse
+ getFeatureInfo(GetFeatureInfo :OA_GetFeatureInfoType) : OA_GetFeatureInfoResponse
+ getLegendGraphic(GetLegendGraphic :OA_GetLegendGraphicType) : OA_GetLegendGraphicResponse
+ getLayerDescription(GetLayerDescription :OA_GetLayerDescriptionType) : OA_GetLayerDescriptionResponse
+ getStyle(GetStyle :OA_GetStyleType) : OA_GetStyleResponse
```

Figure 28. Fundamental workflow in service-driven cartography

The CartoWS is mediating between the data and the cartographic application. It automatically handles all steps necessary for transforming the geodata into a map, while offering to cartographic Web applications a clear interface for accessing and manipulating the representation of the maps.

5.1.2 Cartographic Web Services

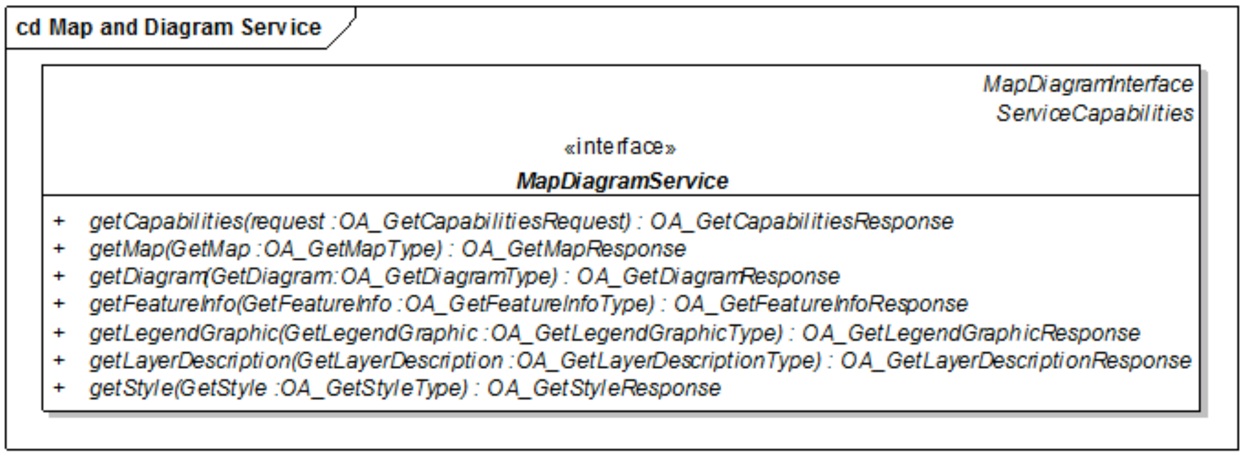
The analysis performed in chapter 3 demonstrated that an approach based on established OGC standards represents a good starting point for this research due to the service-oriented mechanism of the WMS, and the generality of SE and SLD. The WMS (Web Map Service) standard defines a map service as delivering “maps of spatially referenced data dynamically from geographic information”, while the map is defined as a “portrayal of geographic information as a digital image file suitable for display on a computer screen” [ISO, 2005]. For the definition of a Cartographic Web Service (CartoWS) we build upon the existing standards by enhancing the definition of the WMS. The main conceptual difference between the WMS and CartoWS is that the production of the map should be based exclusively on a map representation.

On this basis, we define Cartographic Web Services (CartoWS) as a means for producing maps of spatially referenced data dynamically from geographic information based on an open map representation expressed in SLD & SE. The map representation should define a formal, correct and expressive symbolization of topographic and thematic maps. The map representation that is used for producing the maps should be open source and made available for download.

5.1.3 The Interface of CartoWS

CartoWS can be specified like any other Web Service by its interface and messages. The definition of the CartoWS interface follows and enhances the Web Map Service (WMS) standard. The Map and Diagram Service Interface (MDSI) is the proposed specification for the formal description of the CartoWS concept. The name “Map and Diagram” was set in order to suggest the service functionality for producing both topographic and thematic maps, which is beyond the possibilities of the WMS standard.

Making use of specific computer science knowledge as introduced in chapter 2, such as Web services and their mechanism, concepts and properties, we specify the interface presented in *Figure 29*, which details the operations of CartoWS [Iosifescu, 2007a].



```
+ getCapabilities(request :OA_GetCapabilitiesRequest) : OA_GetCapabilitiesResponse
+ getMap(GetMap :OA_GetMapType) : OA_GetMapResponse
+ getDiagram(GetDiagram:OA_GetDiagramType) : OA_GetDiagramResponse
+ getFeatureInfo(GetFeatureInfo :OA_GetFeatureInfoType) : OA_GetFeatureInfoResponse
+ getLegendGraphic(GetLegendGraphic :OA_GetLegendGraphicType) : OA_GetLegendGraphicResponse
+ getLayerDescription(GetLayerDescription :OA_GetLayerDescriptionType) : OA_GetLayerDescriptionResponse
+ getStyle(GetStyle :OA_GetStyleType) : OA_GetStyleResponse
```

Figure 29. The interface of the Cartographic Web Service [Iosifescu, 2007a]

5.1.4 An Overview of the CartoWS Operations

The operations defined in the Map Diagram Interface are enhancing and completing the operations available in the WMS and SLD standards: the getMap operation returns a map of spatially referenced geographic and thematic information as an image document; the getDiagram operation returns a diagram representation of tabular data as an image document; the getFeatureInfo returns information about the features rendered at a certain point on a map or diagram layer; the getLegendGraphic returns a legend corresponding to a layer as an image document; the getLayerDescription operation returns a layer description document containing the schema information for a layer. In addition, the getStyle operation returns the map representation (referred to as "style" in the OGC terminology) associated with a layer.

5.1.5 The Usefulness of CartoWS Operations

Considering the overall goals of the research, the CartoWS interface must support the development of cartographic applications for the Web.

A clear interface between clients and servers simplifies and decouples the architecture of any Web application, which enables each part to evolve independently. This is a major advantage brought by CartoWS to the architecture of cartographic Web applications.

Moreover, through the defined operations, cartographic web services contribute to the development of cartographic applications. This can be demonstrated in relation with the fundamental functionalities contained by cartographic applications, namely map creation and presentation, the spatial navigations controls, the layer manager, the display of additional information and the display of layer legends.

5.1.6 Usefulness for Map Creation and Spatial Navigation

The GetMap operation, besides providing the mapping core of the application, also assists in the effective creation of spatial navigation controls. The map navigation (zooming, panning, etc.) and the layers displayed by the layer manager can be precisely controlled by the parameters of this request. Therefore when a user zooms or pans a map region, or turns layers on and off, the application must only change the GetMap request parameters to reflect the interactions of the user and the new map view will be generated accordingly by the CartoWS without additional efforts from the application developer. Moreover, the GetMap operation allows a cartographic Web application to define, retrieve or alter a given map representation.

5.1.7 Usefulness for Diagram Creation and Attribute Data Display

The GetDiagram operation allows the display of information as complex diagrams and returns a diagram in the form of an image document. The diagrams can be constructed based on the meta-information retrieved in real time by the GetLayerDescription operation, as this operation offers extended information about the data layer (including the attribute names and corresponding value types and ranges).

The GetFeatureInfo operation is useful for attribute data display. Tooltips are an example of a widely used interactive method for displaying additional information. The GetFeatureInfo allows an application to request additional information by sending the coordinates of the mouse pointer on the map and the layers that compose the map. In a similar manner to WMS, the response of a GetFeatureInfo request will provide all the information available for the geospatial features that were used to create the map region over which the user is holding the mouse.

5.1.8 Usefulness for Legends and User Symbolization

The GetLayerLegend operation used in combination with the GetFeatureInfo, GetStyle and GetMap supports the client-side implementation of smart legends. The GetLayerLegend operation returns a map or diagram legend in the form of an image document for a specific layer and a specific classification. The GetFeatureInfo operation returns information about the features present at/near a specified point in a map, therefore allowing the client to identify and highlight the corresponding legend. GetStyle operation allows the clients to retrieve the map representation (style) associated with any layer offered by the server. Therefore, when the user selects the legend, the cartographic application can perform a GetStyle request, change the map representation for the features represented in the legend, and make a new GetMap request with the modified map representation in order to highlight the features corresponding to the legend.

5.1.9 Usefulness for the Exchange of Data and Map Representation

Besides providing a map representation with a GetMap request, the CartoWS interface enables clients to optionally send data in GML as part of the request message. With the option of sending data directly, a CartoWS allows the creation of clients with more complex interactive functionalities (overlay of user data layers, e.g. visualization of digitized field data). Finally, the use of GetStyle request opens new possibilities for disseminating and reusing the cartographic knowledge. By sharing the formal description of the map representation, the possibility can be envisioned towards a collaborative improvement of the cartographic output.

5.2 Formal Specification of Cartographic Web Services

5.2.1 Specification of the Interface in UML

The interface is specified in detail in the Map and Diagram Service Interface Specifications version 3.0 [Iosifescu, 2007a]. The interface specifications were modeled in Unified Modeling Language (UML) according to the Reference Model for the ORCHESTRA Architecture [OGC, 2007b]. The parameters required by each operation are unambiguously defined through their corresponding UML types presented in *Figure 30*.

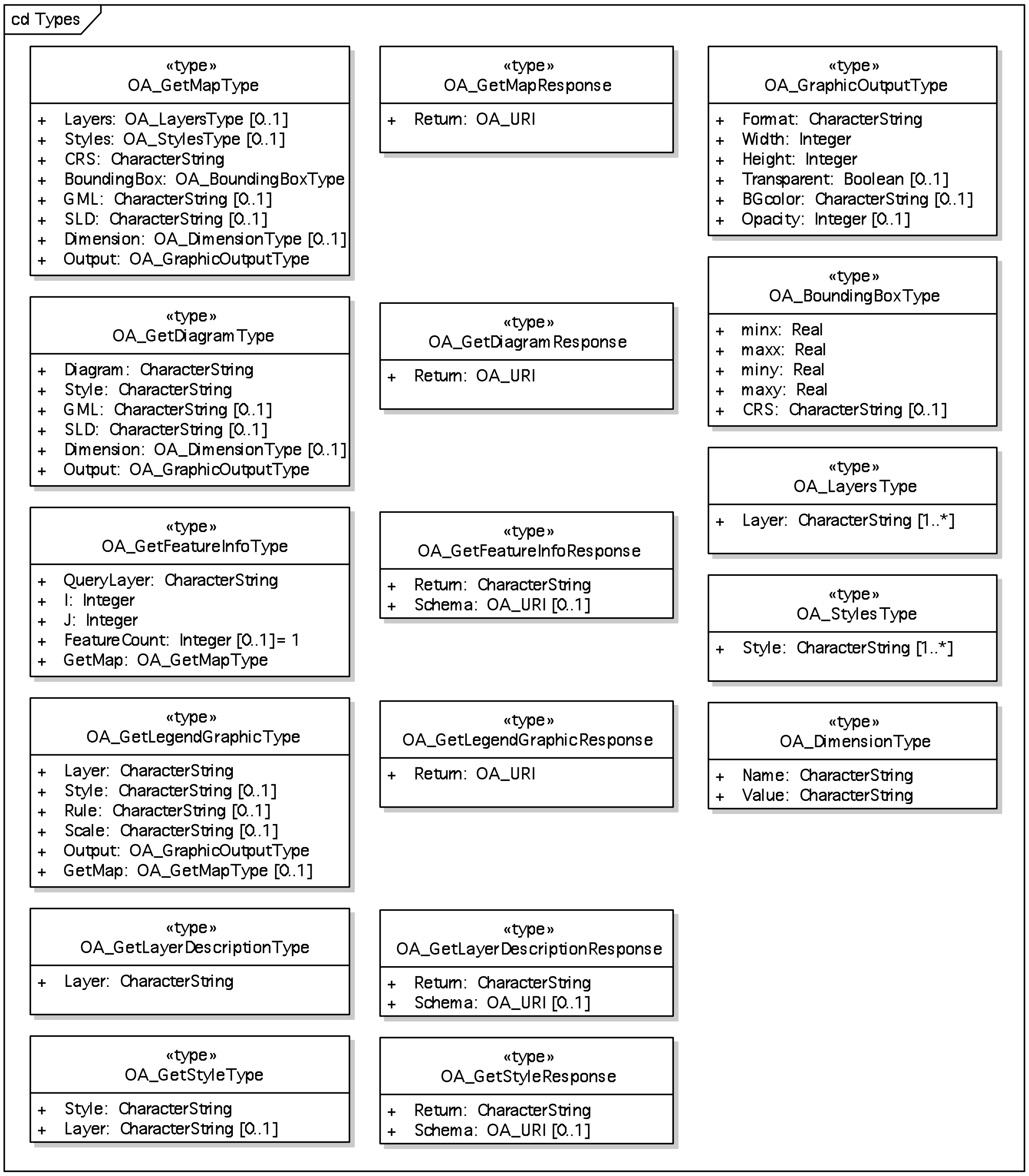

cd Types
«type» OA_GetMapType
+ Layers: OA_LayersType [0..1]
+ Styles: OA_StylesType [0..1]
+ CRS: CharacterString
+ BoundingBox: OA_BoundingBoxType
+ GML: CharacterString [0..1]
+ SLD: CharacterString [0..1]
+ Dimension: OA_DimensionType [0..1]
+ Output: OA_GraphicOutputType
«type» OA_GetMapResponse
+ Return: OA_URI
«type» OA_GraphicOutputType
+ Format: CharacterString
+ Width: Integer
+ Height: Integer
+ Transparent: Boolean [0..1]
+ BGcolor: CharacterString [0..1]
+ Opacity: Integer [0..1]
«type» OA_GetDiagramType
+ Diagram: CharacterString
+ Style: CharacterString
+ GML: CharacterString [0..1]
+ SLD: CharacterString [0..1]
+ Dimension: OA_DimensionType [0..1]
+ Output: OA_GraphicOutputType
«type» OA_GetDiagramResponse
+ Return: OA_URI
«type» OA_BoundingBoxType
+ minx: Real
+ maxx: Real
+ miny: Real
+ maxy: Real
+ CRS: CharacterString [0..1]
«type» OA_LayersType
+ Layer: CharacterString [1..*]
«type» OA_GetFeatureInfoType
+ QueryLayer: CharacterString
+ I: Integer
+ J: Integer
+ FeatureCount: Integer [0..1]= 1
+ GetMap: OA_GetMapType
«type» OA_GetFeatureInfoResponse
+ Return: CharacterString
+ Schema: OA_URI [0..1]
«type» OA_StylesType
+ Style: CharacterString [1..*]
«type» OA_GetLegendGraphicType
+ Layer: CharacterString
+ Style: CharacterString [0..1]
+ Rule: CharacterString [0..1]
+ Scale: CharacterString [0..1]
+ Output: OA_GraphicOutputType
+ GetMap: OA_GetMapType [0..1]
«type» OA_GetLegendGraphicResponse
+ Return: OA_URI
«type» OA_DimensionType
+ Name: CharacterString
+ Value: CharacterString
«type» OA_GetLayerDescriptionType
+ Layer: CharacterString
«type» OA_GetLayerDescriptionResponse
+ Return: CharacterString
+ Schema: OA_URI [0..1]
«type» OA_GetStyleType
+ Style: CharacterString
+ Layer: CharacterString [0..1]
«type» OA_GetStyleResponse
+ Return: CharacterString
+ Schema: OA_URI [0..1]

Figure 30. Overview of the UML types used by the Map and Diagram Service operations

In this figure, each box depicts an UML type used in the definition of the Map and Diagram Interface. The upper part of the box contains the name of the type; the lower part contains its attributes/parameters.

5.2.2 Description of the UML Types

For the sake of completeness, in the following we provide a formal description of the defined UML types, with additional explanations regarding constraints or semantics that cannot be expressed through UML (and therefore not presented in *Figure 30*).

The OA_GetMapType parameter specifies the output image characteristics (format, width, height, transparency, etc.) using the OA_GraphicOutputType as well as a list of layers and their corresponding styles, coordinate reference system, global bounding box and dimension. If map layers and styles are going to be provided using GML and a map representation in SLD, then the Layers and Styles parameters of the operation are optional.

The OA_LayersType parameter references requested layers by using (multiple) Layer character strings. The OA_StylesType parameter references requested styles by using (multiple) Style character strings. The OA_BoundingBoxType parameter specifies an OGC bounding box. The OA_DimensionType parameter selects a name and a value for a layer dimension. Example dimensions could be TIME and ELEVATION.

The OA_GraphicOutputType parameter specifies the characteristics of the image document returned in response to a map or diagram request. These characteristics are format, width, height, transparency, background color and opacity. The OA_GetMapResponse parameter specifies the response of getMap operation. It should contain an URI to the rendered image document.

The OA_GetDiagramType parameter specifies the output image characteristics (format, width, height, transparency, etc.) using the OA_GraphicOutputType as well as a diagram and its corresponding style and optionally layer dimension. If diagram and style are going to be provided using GML and SLD, then the Diagram and Style parameters of the operation are optional. The OA_GetDiagramResponse parameter specifies the response of getDiagram operation. It should contain an URI to the rendered image document.

The OA_GetFeatureInfoType parameter contains the I and J coordinates of the of the query point in the image coordinate system, the layer/diagram name, and the number of features for which is expected to receive information, as well as a copy of the request that generated the image. The image coordinate system is defined in a similar manner with the Cartesian screen coordinate system, with the origin in the left upper corner of the image, x-axis pointing to the right and y-axis pointing down. The x-coordinate of the query point should always be smaller than the requested image width and the y-coordinate should be always smaller than the requested image height. The OA_GetFeatureInfoResponse parameter specifies the response of getFeatureInfo operation. The feature information is returned as character strings and they are accompanied by a document (schema) describing the corresponding type of the returned information.

The OA_GetLegendGraphicType parameter specifies the output image characteristics (format, width, height, transparency, etc.) as well as the layer, style, rule, scale and optionally a copy of the request that generated the image if necessary (if the layer and style are not available on the server). The OA_GetLegendGraphicResponse parameter specifies the response of getLegendGraphic operation. It should contain an URI to the rendered image document. The OA_GetLayerDescriptionType parameter specifies the layer name for which schema information should be returned. The OA_GetLayerDescriptionResponse parameter contains descriptive information for a layer: attribute names, types, units and statistical information (when applicable) like value ranges, max, min, sum, average, and optionally the histogram. This information is returned as character strings and they are accompanied by a document (schema) defining the format of returned information.

Finally, the OA_GetStyleType parameter specifies the name of the style (corresponding to a certain layer) which should be returned. The OA_GetStyleResponse parameter contains the style (map representation) used by the server to portray the specific layer. The style should be returned as SLD/SE.

5.2.3 Mappings of MDSI Operations to REST

The above service interface is defined in an abstract manner, and can therefore be implemented in both SOAP and REST style. For a REST implementation, the parameters of an operation (defined in the corresponding UML type) can be explained in the style of the WMS 1.3 standard, as exemplified by the parameters of the GetDiagram and GetStyle requests in *Figure 31*. The other operations are not listed since they have a correspondence to an existing OGC standard, hence a proven mapping to a REST interface.

Request Parameters for GetStyle	**Mandatory/ Optional**	**Description**
VERSION=1.3.0	M	Request version.
REQUEST=GetStyle	M	Request name.
STYLE=style_name	M	Name of one rendering style.
LAYER=layer_name	M	Name of one layer for which the style is requested.

Request Parameters for GetDiagram	**Mandatory/ Optional**	**Description**
VERSION=1.3.0	M	Request version.
REQUEST=GetDiagram	M	Request name.
DIAGRAM=diagram_name	M	Name of one or more diagram layers.

STYLE=style_name	O	Comma-separated list of one rendering style per requested layer.
WIDTH=output_width	M	Width in pixels of diagram picture.
HEIGHT=output_height	M	Height in pixels of diagram picture.
FORMAT=output_format	M	Output format of diagram.
TRANSPARENT=TRUE|FALSE	O	Background transparency of diagram (default=FALSE).
BGCOLOR=0xFFFFFF	O	Color to be used as the background pixels of the diagram
EXCEPTIONS=exception_format	O	The format in which exceptions are to be reported by the WMS (default=XML).
TIME=time	O	Time value of diagram desired – for temporal diagram sequences.
Other sample dimension(s)	O	Value of other dimensions as appropriate.

Figure 31. GetStyle and GetDiagram operations parameters presented in the manner of the WMS specifications

5.2.4 Mappings of Interface Operations to SOAP

The SOAP style was implemented for the ORCHESTRA and SANY projects, in order to overcome the REST limitations in transporting large SLD and GML messages over the HTTP GET protocol. It was mentioned that the W3C recommendation for Web services is WSDL and SOAP. The Map and Diagram Service provides a SOAP interface for all its operations with the goal of making CartoWS accessible to a large variety of applications requiring integration of mapping components based on dynamic discovery. Therefore, the main interface for the Map and Diagram Service Interface is the SOAP binding using the HTTP POST request method. The REST and SOAP bindings can coexist in the same implementation, since the POST binding is declared as optional in the standard WMS specifications and since the REST operations are only of "GET" nature.

The SOAP binding, together with the service description in WSDL, also tries to follow and improve the latest change proposals of the OGC [OGC, 2004] to offer support for WSDL and SOAP. In a SOAP operation request, all the parameters found in the request method are embedded in the SOAP message as exemplified by the simple GetCapabilities SOAP request:

```
<?xml version="1.0" encoding="UTF-8"?>
    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
            <getCapabilities/>
        </soap:Body>
    </soap:Envelope>
```

In response to this SOAP request the server should reply only with its service metadata. This generic form is useful for the discovery of Web mapping services when they are part of a larger network of geospatial Web services (e.g. Web feature services, Web coverage services, etc.), having various functionalities and capabilities.

A SOAP GetMap request must contain the attributes from the OA_GetMapType. In response to such a request with the characteristics expected from the resulting map, the service would reply with a SOAP response message containing the map as a link to an image file accessible on the Web or with a SOAP attachment. The latter case is illustrated below:

```
<?xml version="1.0" encoding="UTF-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Body>
         <wms:getMapResponse xmlns:wms="http://www.opengis.net/wms">
            <return href="image@mapservice" xsi:type="image/png" />
         </wms:getMapResponse>
   </soap:Body>
</soap:Envelope>
--MIME_boundary
Content-Type: image/png
Content-Transfer-Encoding: binary
Content-ID: <image@mapservice> […image_bytes…]
```

5.2.5 Further Specifications

The complete description in prose, UML, WSDL and XML schema of the Map and Diagram Interface, the interface operations, the operations parameters, and guidelines for the implementation can be found in the ORCHESTRA deliverables [Iosifescu, 2007b]. The in-depth explanation of the formal specifications requires knowledge of several other ORCHESTRA documents, including the Reference Model of the Orchestra Architecture [RMOA, 2007].

The interested reader may refer also to the documents section of [ORCHESTRA Website, 2010] for a complete technical description of the Map and Diagram Service Interface and its role as a Cartographic Web Service in the ORCHESTRA Architecture. For a more compact but less technical overview on the achievements in defining cartographic Web Services, we also recommend the article "Web cartography with open standards", published in the Environmental Modeling and Software journal [Iosifescu et al., 2010a].

5.2.6 CartoWS as a Complete Specification for Service-driven Cartography

The CartoWS interface offers through several operations a communication mechanism between a cartographic application as a service consumer and the CartoWS as service provider. The interface operations are defining the communication method, which represent the actual exchange of messages within the request-response mechanism. The messages are defined through the parameters of a specific operation. The main operation, namely GetMap is also the most complex. The central part of the OA_GetMapType payload is represented by the map representation expressed in the enhanced SLD/SE syntax discussed in the previous chapter.

The definition and specification of the CartoWS is the second piece of the puzzle. CartoWS makes use the detailed syntax for expressing the map representation and enables us to proceed to the next step in our research, namely the proof-concept software implementation of the discussed concepts.

6

Proof-of-Concept Software Implementation

This chapter proves the applicability CartoWS and of the map representation for real world applications. It describes the proof-of-concept implementation of the CartoWS in the QGIS mapserver. The availability of QGIS mapserver proves that the CartoWS concepts can be successfully implemented in software, therefore demonstrating their practical applicability.

6.1 The Software Implementation of CartoWS Specifications

6.1.1 Implementation Concept

The first step in proving the applicability of CartoWS specifications was their successful implementation in software. This software is in fact an enhanced map server, which is capable to generate requested maps based on GIS data and corresponding map representations.

When these requirements for service-based mapping come into play, practical considerations highlight the advantages that existing desktop GIS and mapping software can provide: existing desktop GIS software is natively supporting viewing and analytical functionality for various formats of GIS data. But desktop GIS software was designed to be used locally and thus to render the GIS data to the screen of the local workstation.

The implementation of the CartoWS specifications in software, started from the hypothesis that desktop GIS software can be enhanced in such a manner that it becomes accessible as a service over the Web [Hugentobler et al., 2007]. As a consequence, such an implementation is based on the libraries and functionalities of a GIS system, thus containing GIS viewing and analytical functionality.

Finally, three major changes of the desktop GIS have been identified in order to carry out the Web enablement of desktop GIS Libraries. They are the transformation of GUI-based user input into a map representation (SLD/SE), the transformation of the GUI based rendering into an in-memory map rendering, and the provision of the software with the corresponding Web interface (WMS/MDSI).

6.1.2 Proof-of-Concept Implementation in Software

The proof-of-concept implementation in software is QGIS mapserver. The QGIS mapserver is implemented in C++ as a FastCGI/CGI application. It works together with a Web server such as Apache that is able to invoke it as a FastCGI/CGI dynamic application. The Web server receives a request from a Web client, e.g. a Web browser, a client side script-based application or a desktop GIS such as QGIS, uDig or JUMP. The Web server then passes the calls to the map server application, whereas the calling mechanism is dependant on the used server technology, which in this case is FastCGI/CGI.

QGIS mapserver was created to prove the improved cartographic output as defined in this thesis. In comparison with other map server implementations, QGIS mapserver uses the enhanced SLD/SE syntax presented in chapter 4 as a native map representation language for predefined configuration of hosted layers and styles, as well as for user defined layers and styles. Moreover, it is compatible with the OGC WMS (1.3.0 and 1.1.1) specifications since they are a subset of the CartoWS specifications.

6.1.3 Technological Foundations for the Implementation

The prototype implementation of CartoWS in software is based on the libraries of the open source Quantum GIS and it consequently influenced the name of the implementation as "QGIS mapserver" [QGIS mapserver Website, 2010]. QGIS is an open source desktop GIS with user-friendly graphical user interface that provides viewing, editing, printing and also analytical functionality for geodata [QGIS, 2010]. The QGIS code modular and non-GUI functionality is compiled into a core library that can be used by any application, provided it is licensed under the GNU GPL. This modularity led to the decision to write the implementation of CartoWS based on desktop GIS rather than on an existing mapserver [Hugentobler et al., 2007].

6.1.4 Basic Mapserver Design

The software design of QGIS mapserver reflects the implementation concept explained at the beginning of the chapter. Although the implementation evolved over the years, its basic design is shown in the UML class diagram from *Figure 32*.

From the UML diagram it can be observed that QGIS mapserver is composed of the following main classes: a WMSServer class, a RequestHandler class and a SLDParser class.

The WMSServer class controls the execution of the different request types and implements the CGI communication with the underlying Web server that is dispatching the incoming requests. The RequestHandler class parses incoming requests and effectively makes the server code independent of the protocol used. Examples of subclasses are handlers for HTTP GET, HTTP POST or SOAP. The class SLDParser provides an entry point to create layer and style objects from the map representation expressed in SLD.

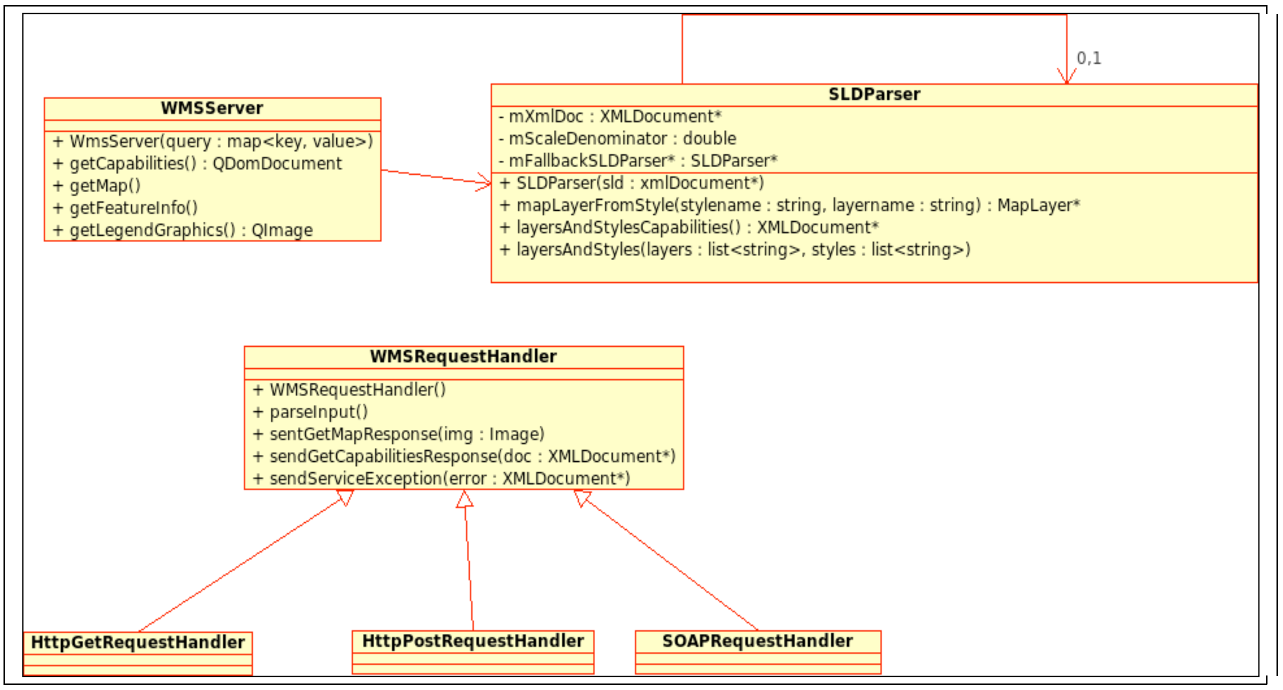

WMSServer
+ WmsServer(query : map<key, value>)
+ getCapabilities() : QDomDocument
+ getMap()
+ getFeatureInfo()
+ getLegendGraphics() : QImage
SLDParser
- mXmlDoc : XMLDocument*
- mScaleDenominator : double
- mFallbackSLDParser* : SLDParser*
+ SLDParser(sld : xmlDocument*)
+ mapLayerFromStyle(stylename : string, layername : string) : MapLayer*
+ layersAndStylesCapabilities() : XMLDocument*
+ layersAndStyles(layers : list<string>, styles : list<string>)
0,1
WMSRequestHandler
+ WMSRequestHandler()
+ parseInput()
+ sentGetMapResponse(img : Image)
+ sendGetCapabilitiesResponse(doc : XMLDocument*)
+ sendServiceException(error : XMLDocument*)
HttpGetRequestHandler
HttpPostRequestHandler
SOAPRequestHandler

Figure 32. Basic Software Design of the QGIS mapserver [Hugentobler et al., 2007]

6.1.5 Parsing of the Map Representation

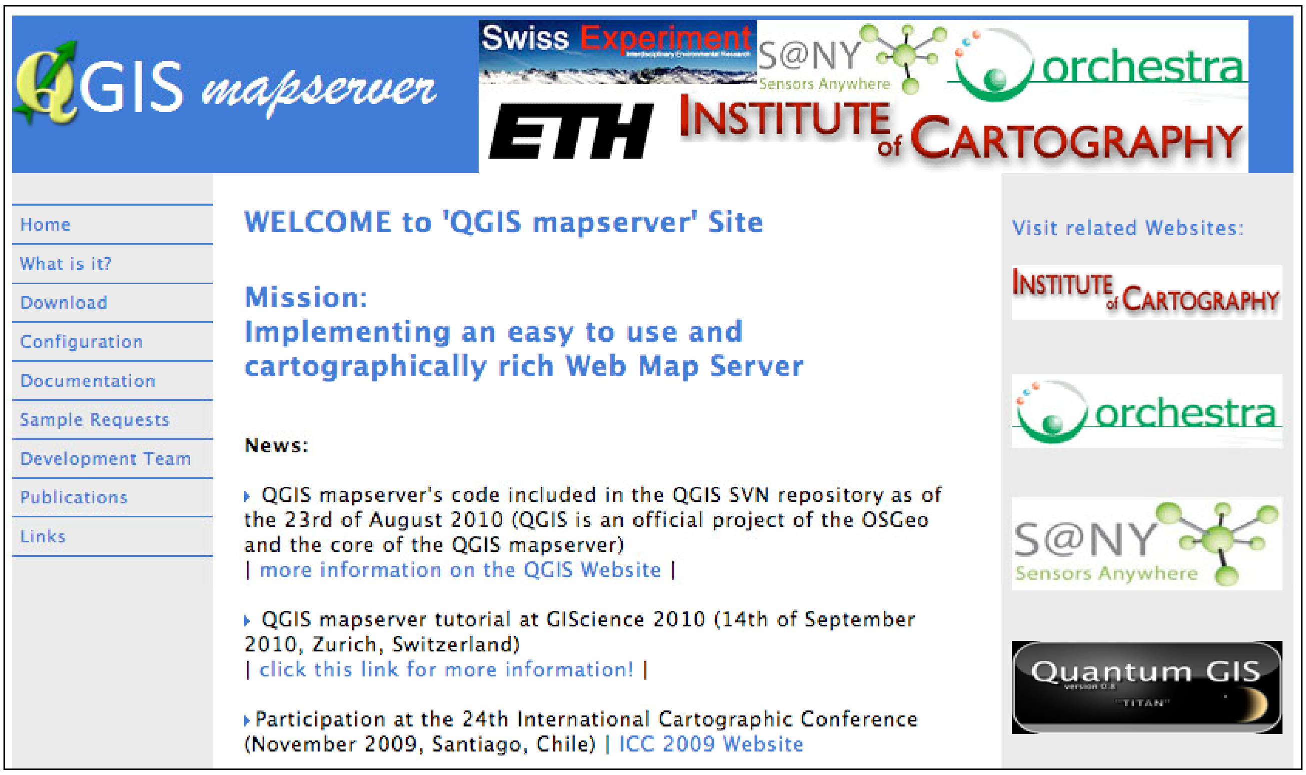
The logical parsing and rendering of the map representation is handled in the QGIS mapserver implementation by important classes such as the QgsLayerBuilder class, the QgsSLDRenderer class, eth QgsSLDRule class and the QgsFilter class. The QgsLayerBuilder is a base class that abstracts the creation of different layer types. Finally, the QgsSLDRule, QgsFilter, and QgsSLDRenderer classes are handling, as their names suggest, the application of the detailed feature symbolization to individual features for rendering, depending of the defined matching conditions.

In the map representation, a layer may have multiple SE FeatureTypeStyle elements with SE rules and matching conditions that coordinate a more complex symbolization. This structure is also reflected in the QGIS mapserver class hierarchy implementation. QgsSLDRenderer contains the vector symbology and describes how a feature is to be rendered. An instance of QgsSLDRenderer may have one or more associated QgsSLDRule objects. For every feature that is to be rendered, QGIS mapserver calls the method QgsSLDRenderer::renderFeature. Inside this method, QgsSLDRenderer goes through the list of rules until it finds the first one that is valid for the feature and applies the corresponding QgsRule object. Then the process is repeated for multi-rule symbolization of the current layer and further for all layers contained in the map representation.

6.1.6 Open Source Software Project

The QGIS map server is an open source software project with a dedicated project Website. As shown in *Figure 33,* the QGIS mapserver homepage offered basic information about the project.

The homepage of the QGIS mapserver also contained the project’s documentation and user manual, it hosted the download of binary packages and source code, and it hosted the repository for the source code. Since mid 2010 QGIS mapserver is included in the official QGIS Desktop distribution, therefore ensuring the future and further development of the QGIS mapserver software [QGIS mapserver Website, 2010].



WELCOME to 'QGIS mapserver' Site

Mission:
Implementing an easy to use and cartographically rich Web Map Server

News:

▸ QGIS mapserver's code included in the QGIS SVN repository as of the 23rd of August 2010 (QGIS is an official project of the OSGeo and the core of the QGIS mapserver)
| more information on the QGIS Website |

▸ QGIS mapserver tutorial at GIScience 2010 (14th of September 2010, Zurich, Switzerland)
| click this link for more information! |

▸ Participation at the 24th International Cartographic Conference (November 2009, Santiago, Chile) | ICC 2009 Website

Figure 33. The homepage of QGIS mapserver [QGIS mapserver Website, 2010]

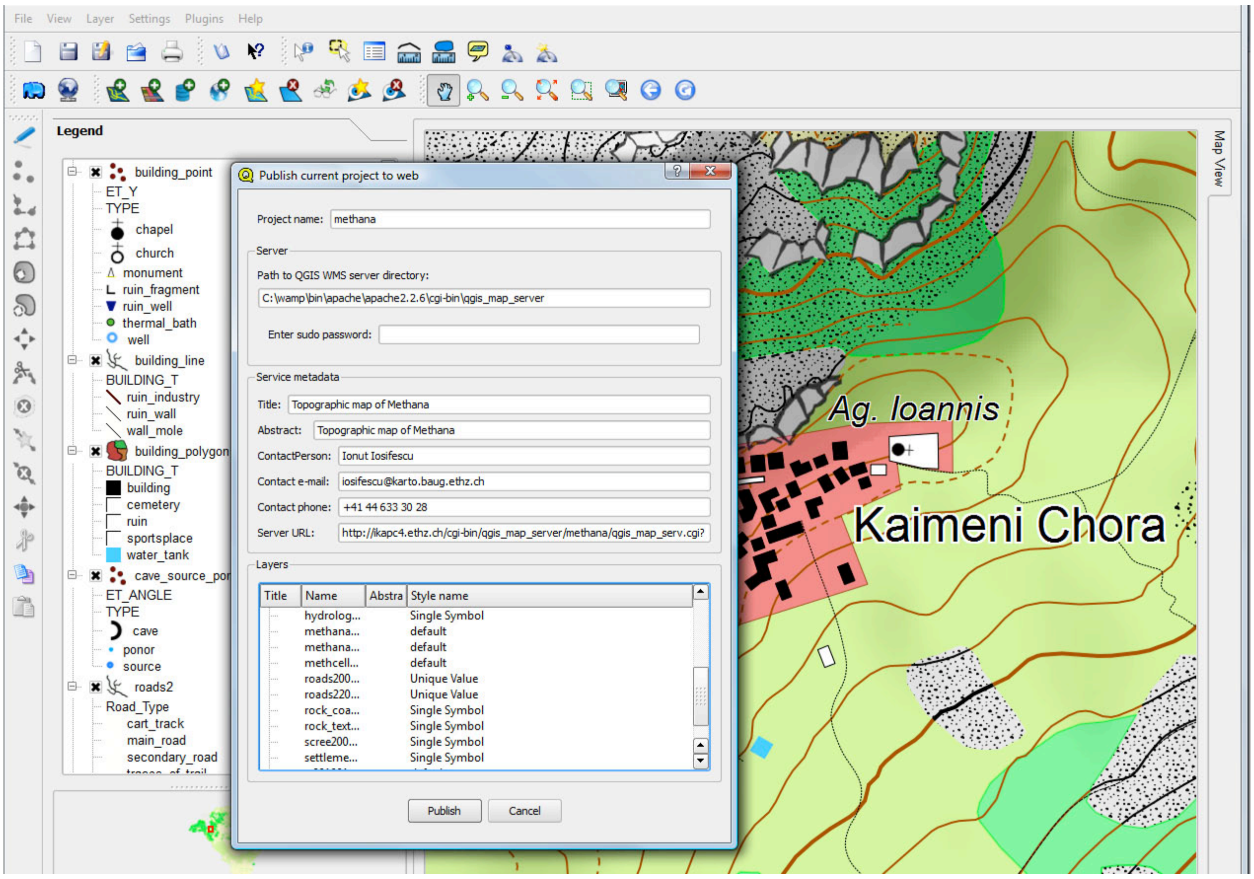
6.1.7 Software Support for Authoring of Map Representations

Although the map representation is human readable and can be edited with a text or XML editor, expressing a complex map representation can become cumbersome due to the length of the XML syntax. For this reason, a graphical configuration of the map representation was designed to further improve the user friendliness as well as the straightforward setup and maintenance of CartoWS.

As geodata administrators typically use desktop GIS software to view and edit geodata prior to making it available over the Web, we have designed the 'Publish to Web' plugin as shown in *Figure 34*.

It is a plugin for QGIS desktop, which exports the current layers and symbologies as a Web project containing the map representation for QGIS mapserver. As QGIS desktop and QGIS mapserver use the same base libraries, the maps that will be created with the map representation look the same as when they are symbolized interactively in QGIS Desktop.

The main advantage of the Publish to Web plugin is that it generates the map representation based on user interaction within the graphical user interface of QGIS. It is limited to basic symbolization as supported by the user interface of the QGIS but the generated map representation offers a structure for manual enhancements.


File View Layer Settings Plugins Help
Legend
Map View
Ag. Ioannis
Kaimeni Chora
Publish current project to web
Publish
Cancel

Project name: methana

Server

Path to QGIS WMS server directory: C:\wamp\bin\apache\apache2.2.6\cgi-bin\qgis_map_server

Enter sudo password:

Service metadata

Title: Topographic map of Methana

Abstract: Topographic map of Methana

ContactPerson: Ionut Iosifescu

Contact e-mail: iosifescu@karto.baug.ethz.ch

Contact phone: +41 44 633 30 28

Server URL: http://ikapc4.ethz.ch/cgi-bin/qgis_map_server/methana/qgis_map_serv.cgi?

Layers

Title	Name	Abstra	Style name
	hydrolog...		Single Symbol
	methana...		default
	methana...		default
	methcell...		default
	roads200...		Unique Value
	roads220...		Unique Value
	rock_coa...		Single Symbol
	rock_text...		Single Symbol
	scree200...		Single Symbol
	settleme...		Single Symbol

Figure 34. The Publish to Web plugin

The long-term vision for the Publish to Web plugin is to make the whole Web technology transparent for any user of QGIS mapserver, specialists or layman alike. Ideally, a user of the Publish to Web plugin should not need to install, configure and know about Web servers, FastCGI, location and arrangement of configuration files and links. The user can then concentrate on the cartographic symbolization of geodata and then let the plugin help at the translation into formal map representations, while QGIS mapserver does the on-demand publishing of the maps on the Web.

6.2 Basic Usage of the Software Implementation

6.2.1 QGIS mapserver set-up

As previously mentioned, the QGIS mapserver is a CGI/FastCGI application. As a result, it requires the availability of a Web server. The installation requires the placement of the QGIS mapserver in the CGI-BIN/FCGI-BIN directories of the Web server. In addition, it requires the addition of the QGIS libraries and dependencies in the PATH environment variable or in the configuration of the Web server. For example, on a Linux operating system, we can configure the QGIS mapserver libraries and dependencies in the Web server by entering the following line into the fcgid.conf configuration file:

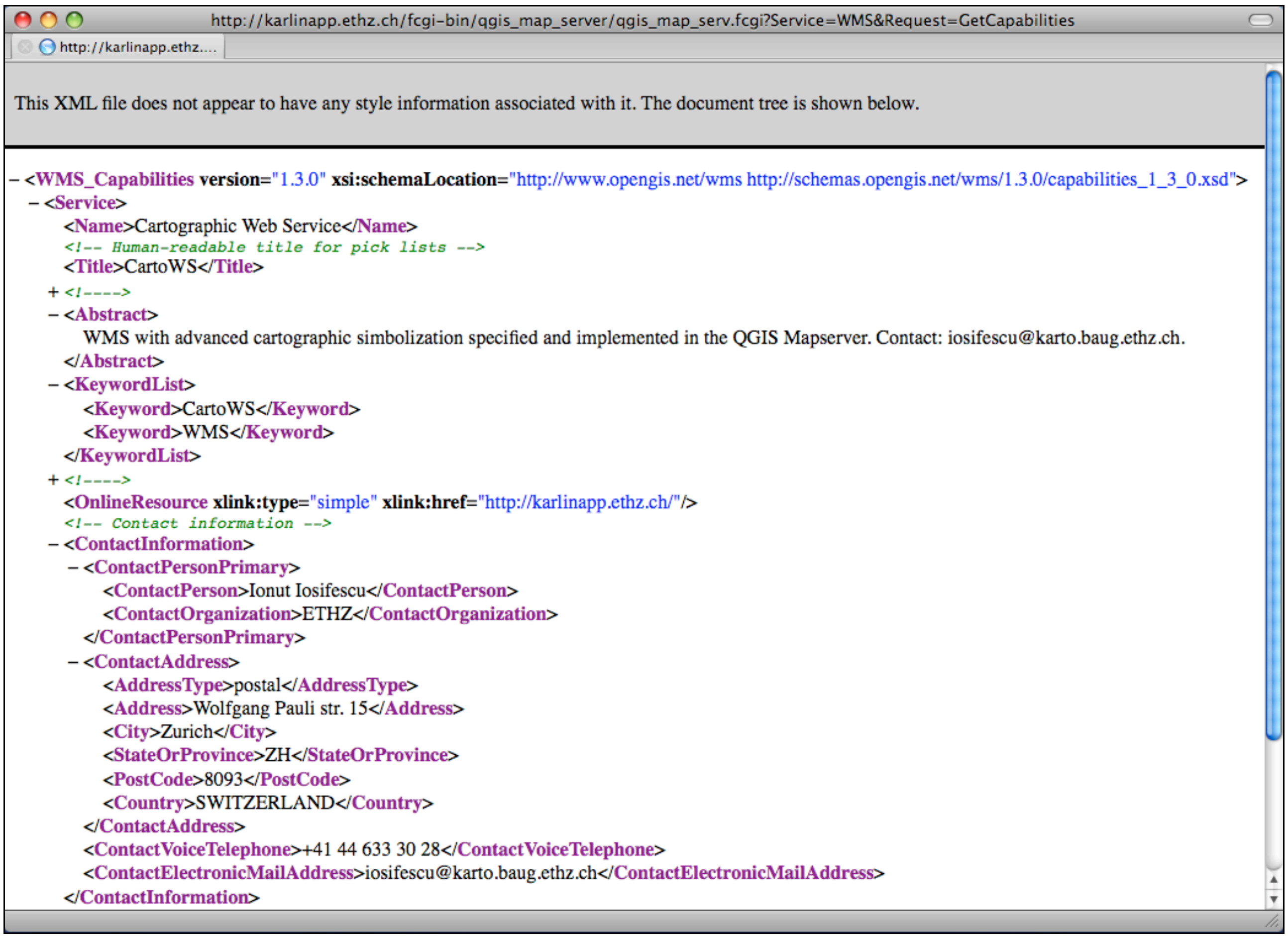
"DefaultInitEnv LD_LIBRARY_PATH <Path to Qt libraries>:<Path to QGIS libraries>".

The installation can be verified by entering the following in a web browser one of the following two requests (depending on the CGI/FCGI configuration):

http://localhost/cgi-bin/qgis_map_server/qgis_map_serv.cgi?SERVICE=WMS&
REQUEST=GetCapabilities&Version=1.3.0

http://localhost/fcgi-bin/qgis_map_server/qgis_map_serv.fcgi?SERVICE=WMS&
REQUEST=GetCapabilities&Version=1.3.0

A successful configuration will be rewarded by a similar WMS capabilities document as shown in the next figure:


http://karlinapp.ethz.ch/fcgi-bin/qgis_map_server/qgis_map_serv.fcgi?Service=WMS&Request=GetCapabilities
http://karlinapp.ethz....

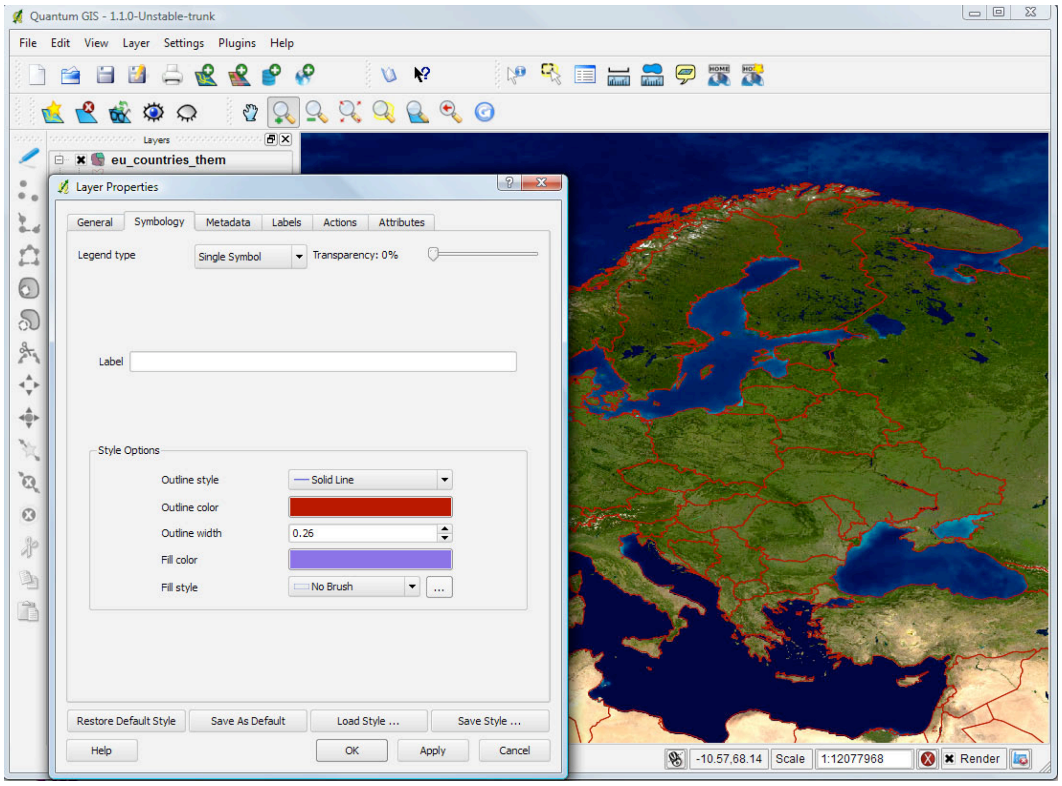
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
– <WMS_Capabilities version="1.3.0" xsi:schemaLocation="http://www.opengis.net/wms http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd">
  – <Service>
      <Name>Cartographic Web Service</Name>
      <!-- Human-readable title for pick lists -->
      <Title>CartoWS</Title>
    + <!---->
    – <Abstract>
        WMS with advanced cartographic simbolization specified and implemented in the QGIS Mapserver. Contact: iosifescu@karto.baug.ethz.ch.
      </Abstract>
    – <KeywordList>
        <Keyword>CartoWS</Keyword>
        <Keyword>WMS</Keyword>
      </KeywordList>
    + <!---->
      <OnlineResource xlink:type="simple" xlink:href="http://karlinapp.ethz.ch/"/>
      <!-- Contact information -->
    – <ContactInformation>
      – <ContactPersonPrimary>
          <ContactPerson>Ionut Iosifescu</ContactPerson>
          <ContactOrganization>ETHZ</ContactOrganization>
        </ContactPersonPrimary>
      – <ContactAddress>
          <AddressType>postal</AddressType>
          <Address>Wolfgang Pauli str. 15</Address>
          <City>Zurich</City>
          <StateOrProvince>ZH</StateOrProvince>
          <PostCode>8093</PostCode>
          <Country>SWITZERLAND</Country>
        </ContactAddress>
        <ContactVoiceTelephone>+41 44 633 30 28</ContactVoiceTelephone>
        <ContactElectronicMailAddress>iosifescu@karto.baug.ethz.ch</ContactElectronicMailAddress>
      </ContactInformation>
```

Figure 35. The default QGIS mapserver capabilities response

6.2.2 Basic Layer Symbolization with QGIS Desktop

The first step necessary in order to generate the structure of the map representation is to install the QGIS Desktop software and the Publish to Web plugin. Then the QGIS Desktop will be used to symbolize similar GIS data to the one we want to create the map representation for. The symbolization will be performed in the “Symbology” tab of the QGIS Graphical User Interface (GUI) as shown in the next figure:


Quantum GIS - 1.1.0-Unstable-trunk
File Edit View Layer Settings Plugins Help
Layers
Layer Properties
General Symbology Metadata Labels Actions Attributes
Restore Default Style
Save As Default
Load Style ...
Save Style ...
Help
OK
Apply
Cancel
Scale
Render

eu_countries_them

Legend type: Single Symbol
Transparency: 0%
Label:

Style Options

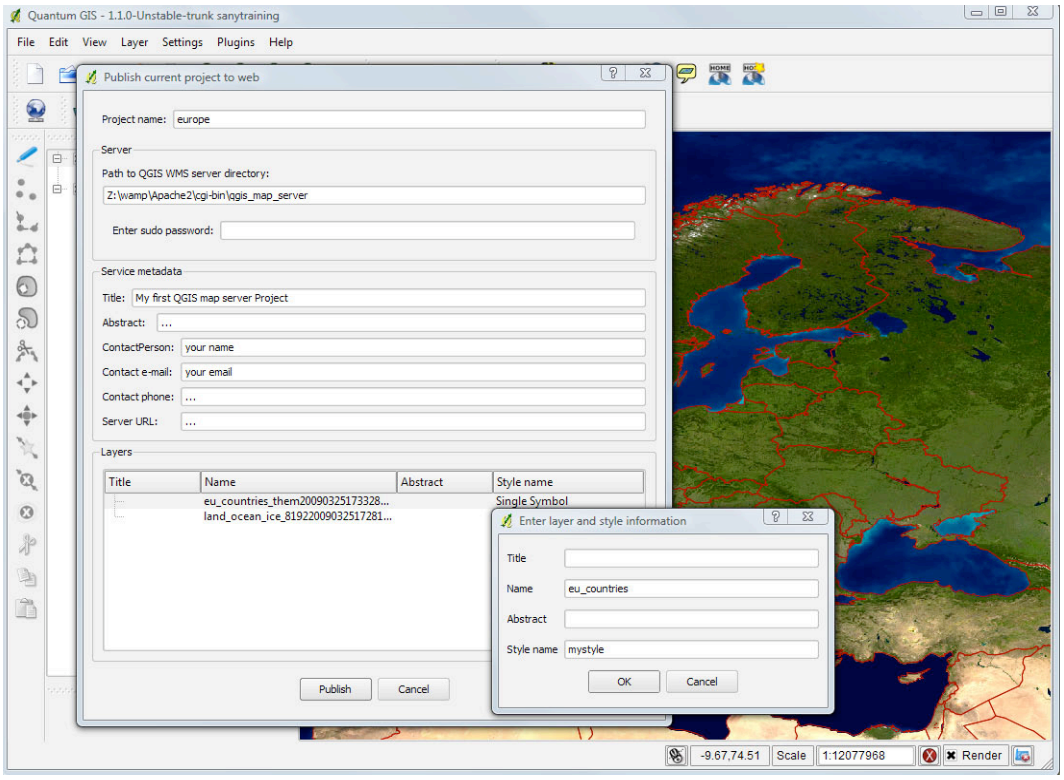
Outline style	Solid Line
Outline color	🟥
Outline width	0.26
Fill color	🟪
Fill style	No Brush

-10.57,68.14 Scale 1:12077968

Figure 36. Symbolizing GIS data with QGIS desktop

6.2.3 Generation of the basic Map Representation

When a user finishes the symbolization of the GIS datasets, the symbolization can be used to generate the map representation corresponding to the datasets. This is achieved by the Publish to Web plugin as illustrated below:


Quantum GIS - 1.1.0-Unstable-trunk sanytraining
File Edit View Layer Settings Plugins Help
Publish current project to web
Publish
Cancel
Enter layer and style information
OK
Cancel
Scale
Render

Project name: europe

Server

Path to QGIS WMS server directory:
Z:\wamp\Apache2\cgi-bin\qgis_map_server

Enter sudo password:

Service metadata

Title:	My first QGIS map server Project
Abstract:	...
ContactPerson:	your name
Contact e-mail:	your email
Contact phone:	...
Server URL:	...

Layers

Title	Name	Abstract	Style name
	eu_countries_them20090325173328...		Single Symbol
	land_ocean_ice_81922009032517281...		

Title	
Name	eu_countries
Abstract	
Style name	mystyle

-9.67,74.51 Scale 1:12077968

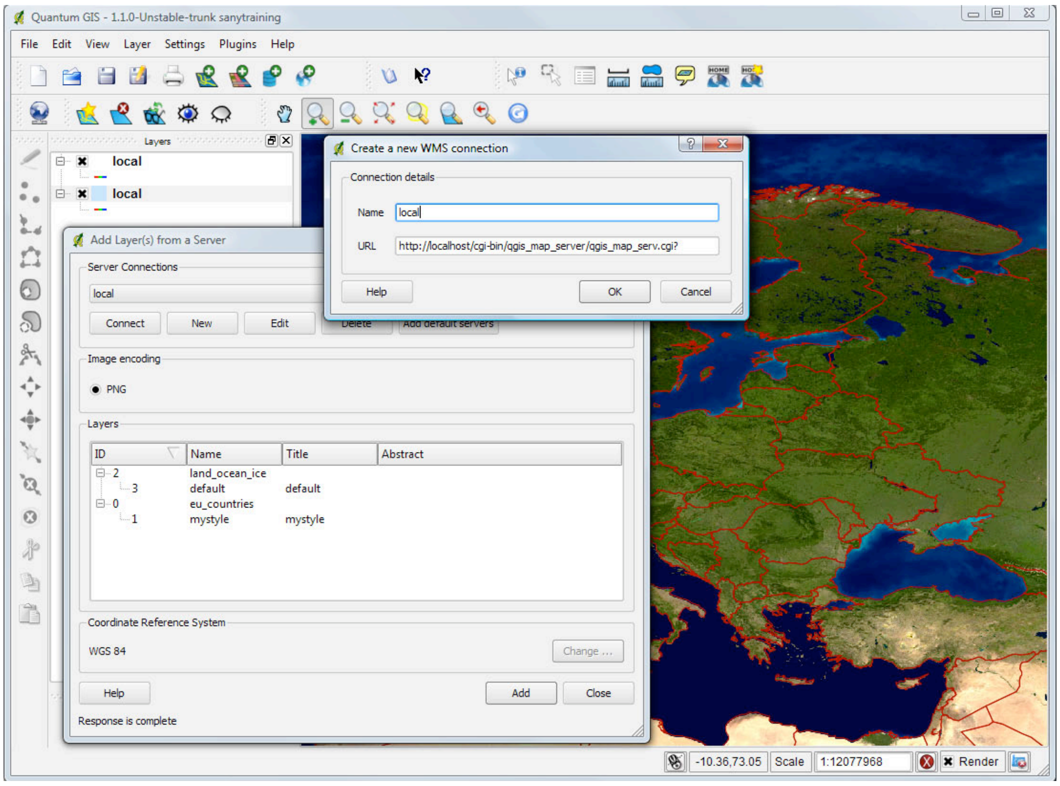
Figure 37. Exporting the map representation with the Publish to Web plugin

This plugin will open a dialog where it is possible to insert metadata, information about the server, layer and style names. The map representation is generated by clicking the "Publish" button.

The QGIS mapserver is configured through a map representation. If the path to the QGIS mapserver CGI-BIN or FCGI-BIN directory is entered correctly, the generated admin.sld file (containing the map representation) is configuring the server with the map output described in the map representation.

This can be tested using the QGIS desktop as WMS client and connecting to the local QGIS mapserver URL: either http://localhost/cgi-bin/qgis_map_server/project_name/qgis_map_serv.cgi? or http://localhost/fcgi-bin/qgis_map_server/project_name/qgis_map_serv.fcgi?

A successful configuration of a Web project will be rewarded with the map produced by the QGIS mapserver from the map representation. The returned image should be identical to the symbolization of the GIS data that was used to generate the map representation:


Quantum GIS - 1.1.0-Unstable-trunk sanytraining
File Edit View Layer Settings Plugins Help
Layers
local
local
Create a new WMS connection
Help
OK
Cancel
Add Layer(s) from a Server
Connect
New
Edit
Delete
Add default servers
Change ...
Help
Add
Close
Render

Connection details
Name: local
URL: http://localhost/cgi-bin/qgis_map_server/qgis_map_serv.cgi?

Server Connections: local

Image encoding: ● PNG

Layers

ID	Name	Title	Abstract
2	land_ocean_ice		
3	default	default	
0	eu_countries		
1	mystyle	mystyle	

Coordinate Reference System: WGS 84

Response is complete

-10.36,73.05 Scale 1:12077968

Figure 38. Testing the correct functionality of a QGIS mapserver project with QGIS Desktop

6.2.4 Integration in a Web Client

The CartoWS specifications and, as consequence, the QGIS mapserver implementation are fully compatible with the WMS standard. Therefore, the maps generated based on the map representation can be integrated in a variety of Web clients that support this standard. The exact procedure depends on the used framework and generally consists in configuring the URL of the mapserver (e.g. http://localhost/cgi-bin/qgis_map_server/qgis_map_serv.cgi?) and the various request parameters such as STYLES and LAYERS. A demonstration for the integration in the OpenLayers client framework is presented in the following figure:

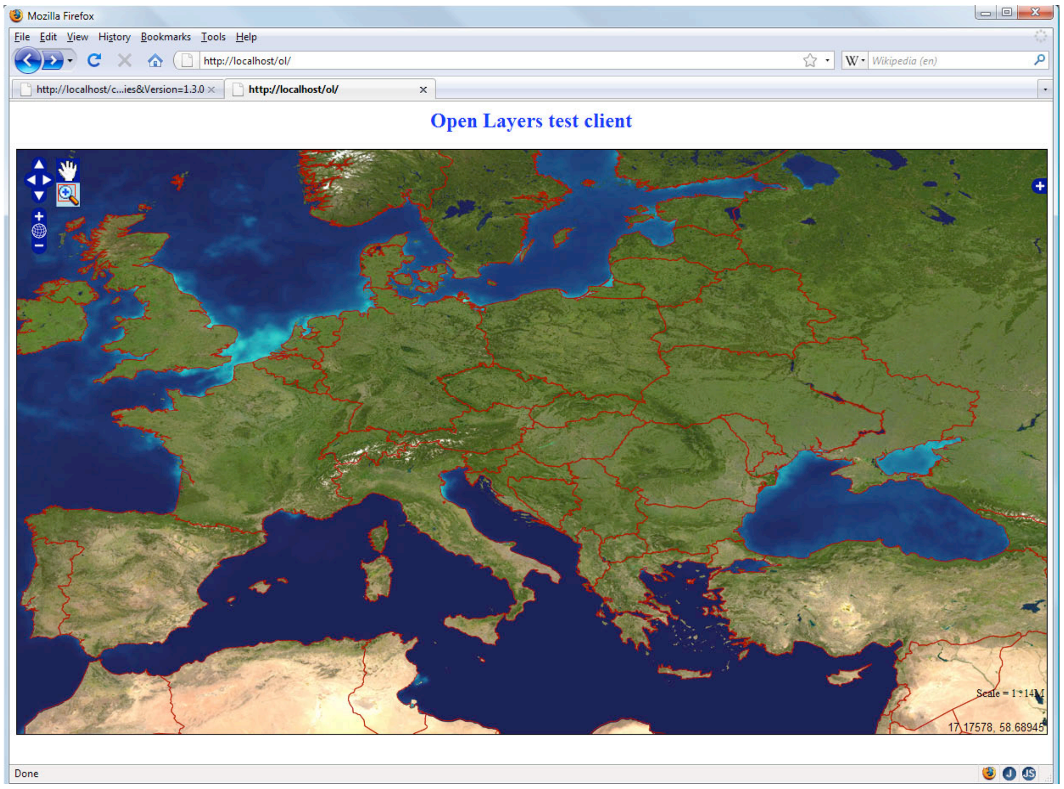
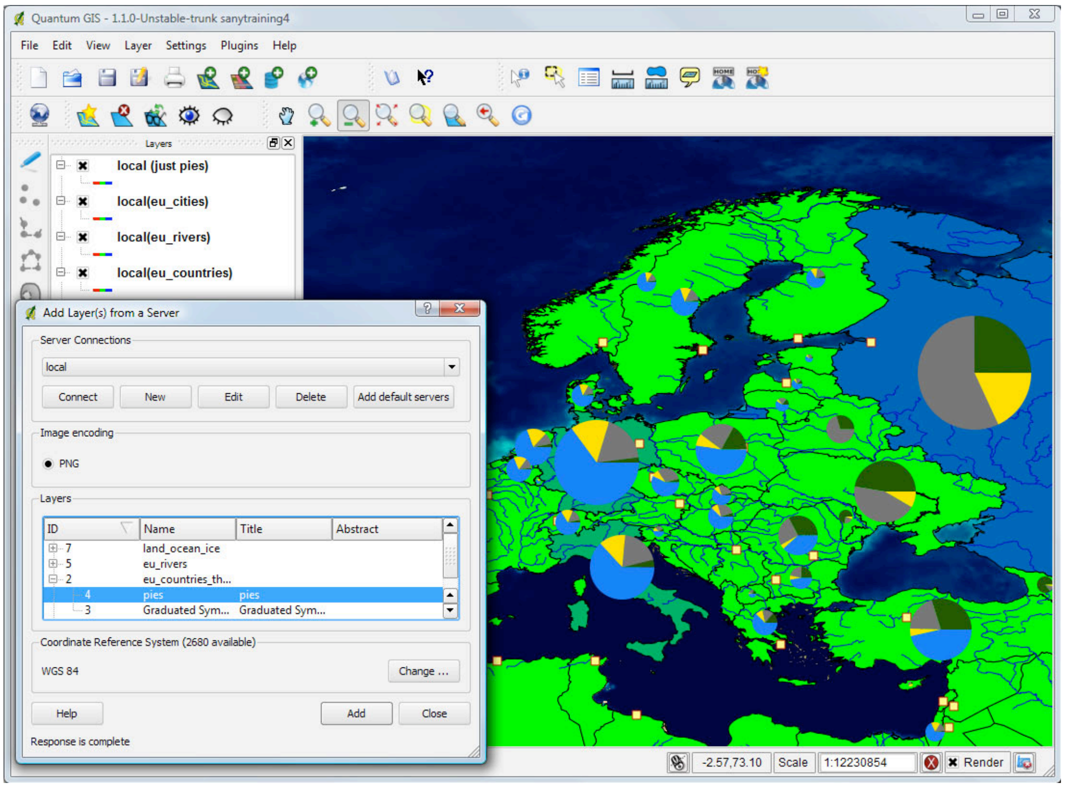


Figure 39. Example integration of maps generated by QGIS mapserver in OpenLayers

6.2.5 Generation of a Map Representation for Thematic Maps

With a similar GUI-based workflow as presented above, a user can also create thematic maps such as choropleth maps and proportional symbol map as shown in *Figure 40* [Iosifescu et al., 2010b].



Server Connections: local

Image encoding: ● PNG

Layers

ID	Name	Title	Abstract
7	land_ocean_ice		
5	eu_rivers		
2	eu_countries_th...		
4	pies	pies	
3	Graduated Sym...	Graduated Sym...	

Coordinate Reference System (2680 available): WGS 84

Response is complete

Figure 40. GUI-based configuration and testing of thematic maps

6.2.6 Editing the Map Representation

The generated map representation can be enhanced manually even further with more expressive symbolization possibilities offered by the map representation (and implemented in the QGIS mapserver) that are not configurable via the QGIS Desktop GUI. Examples of such features are the definitions of Filters for the highlighting of individual point, line and polygon features, the definition of several representation additional options for thematic maps, and the definition of patterns and gradients as shown in the next illustration:

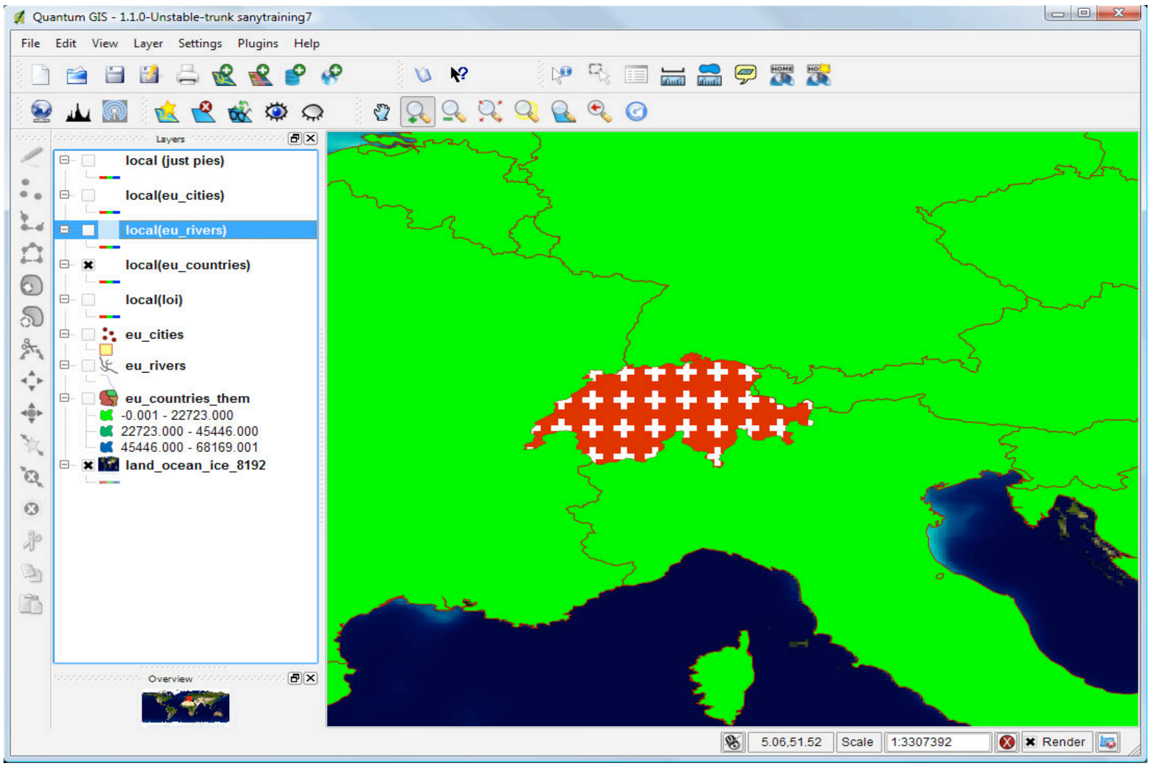


Figure 41. Testing of a Fill pattern from a map representation defined manually

6.2.7 Available User Software Documentation

Detailed documentation and training materials for the configuration and use of the QGIS mapserver can be found the QGIS Project Website. Among various sources we mention the QGIS mapserver manual [Zeder, 2009b] and a compact tutorial organized for the GISScience 2010 conference [Iosifescu et al., 2010b]. Furthermore, additional information can be found in various student works from ETH Zurich such as [Eberle, 2009] and [Zeder, 2009a].

6.3 Software Foundations for the Development of Cartographic Applications

The existence of a mapserver that is using the map representation for cartography proves that the CartoWS concepts and specifications can be successfully implemented in software, therefore demonstrating their practical applicability.

The QGIS mapserver, according to the CartoWS concepts, is mediating between the data and a cartographic application. QGIS mapserver automatically handles all steps necessary for transforming the geodata into a map, while offering to cartographic Web applications a clear interface for accessing the maps and their map representation. Therefore, it offers new possibilities for the implementation of Web cartographic applications that are always up-to-date, expressive and interactive.

7 Proof-of-Concept Applications

This chapter presents three cartographic Web applications namely the SANY Decision Support System, the COGEAR GIS Platform for Environmental Research and the Web-based GIS software. These applications are used in order to show the applicability of CartoWS concepts under real world conditions. The advantages of the service-driven cartography based on map representations are visible in useful, up-to-date, expressive and interactive real-world applications.

7.1 Up-to-Date and Interactive Maps

7.1.1 The SANY Decision Support System

The proof for the suitability and applicability of the CartoWS concepts in real-world cartographic products is demonstrated with several examples of useful applications. The first example is the SANY Decision Support Infrastructure [Jaques, 2007; Klopfer et al., 2009].

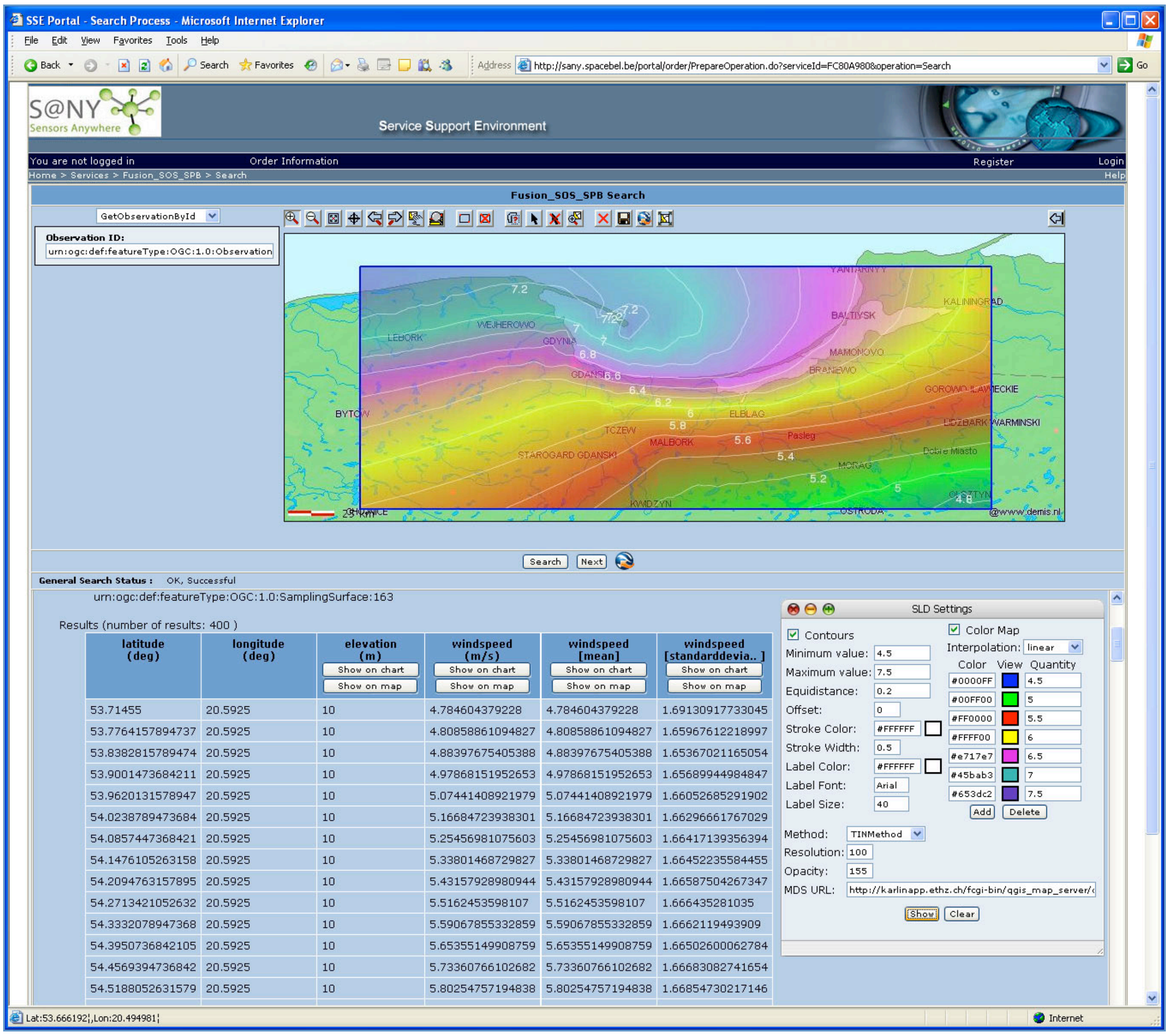
The SANY Decision Support System (DSS) consists mainly of a Web portal providing to decision makers several main functionalities such as the discovery of sensor related services, standard access to sensor observations from different providers, starting and monitoring of fusion services acting on sensor data and the visualization of sensor data on geographic maps [Jaques, 2007].

7.1.2 The Design of the Web Mapping Component

In the SANY DSS design of the Web mapping component, cartographic web services play the central role. From the cartographic point of view, the SANY DSS is displaying maps of sensor fusion (e.g. a spatial or spatio-temporal interpolation) in near-real time. This is achieved with a Web-mapping component that abides the reference workflow for service-driven cartography discussed in chapter 5 and graphically summarized in *Figure 28*. Specifically, the geodata consists of real-time sensor data sources or results of fusion processes operated on sensor data and the map representation specifies the visualization of the source data on the map with quantitative area symbolization and isolines. The maps are generated on demand based on the map representation by the prototype software implementation of CartoWS, namely the QGIS mapserver and finally the SANY DSS map viewer is the front-end (client) for presenting the maps and interacting with the users.

7.1.3 Cartographic Functionalities of the SANY DSS

The QGIS mapserver (as implementation of CartoWS concepts and specifications) enables the DSS application to offer complex client-side functionalities to their users, i.e. real time user editing or display of processed data, such as interpolated wind direction and wind speed. In the case of the SANY DSS, it is possible to use diverse data sources such as fusion results available on FTP server or another Web service like a Sensor Observation Service (SOS) or to directly embed the computed data inside the request. The CartoWS encourages the clients to send their data for map rendering using open OGC standards, such as GML, or in the Sensor Web Observation and Measurements XML format. The results are highly interactive and customizable on-demand maps presented through the user interface shown in *Figure 42*. For example, the visualization of wind direction and speed in map form is important for the simulation of the dispersion of airborne pollutants and therefore provides valuable input for the decision processes related to environmental air quality.



Observation ID: urn:ogc:def:featureType:OGC:1.0:Observation

General Search Status : OK, Successful

urn:ogc:def:featureType:OGC:1.0:SamplingSurface:163

Results (number of results: 400)

latitude (deg)	longitude (deg)	elevation (m)	windspeed (m/s)	windspeed [mean]	windspeed [standarddevia..]
53.71455	20.5925	10	4.784604379228	4.784604379228	1.69130917733045
53.7764157894737	20.5925	10	4.80858861094827	4.80858861094827	1.65967612218997
53.8382815789474	20.5925	10	4.88397675405388	4.88397675405388	1.65367021165054
53.9001473684211	20.5925	10	4.97868151952653	4.97868151952653	1.65689944984847
53.9620131578947	20.5925	10	5.07441408921979	5.07441408921979	1.66052685291902
54.0238789473684	20.5925	10	5.16684723938301	5.16684723938301	1.66296661767029
54.0857447368421	20.5925	10	5.25456981075603	5.25456981075603	1.66417139356394
54.1476105263158	20.5925	10	5.33801468729827	5.33801468729827	1.66452235584455
54.2094763157895	20.5925	10	5.43157928980944	5.43157928980944	1.66587504267347
54.2713421052632	20.5925	10	5.5162453598107	5.5162453598107	1.666435281035
54.3332078947368	20.5925	10	5.59067855332859	5.59067855332859	1.6662119493909
54.3950736842105	20.5925	10	5.65355149908759	5.65355149908759	1.66502600062784
54.4569394736842	20.5925	10	5.73360766102682	5.73360766102682	1.66683082741654
54.5188052631579	20.5925	10	5.80254757194838	5.80254757194838	1.66854730217146

SLD Settings

✓ Contours

Minimum value:	4.5
Maximum value:	7.5
Equidistance:	0.2
Offset:	0
Stroke Color:	#FFFFFF
Stroke Width:	0.5
Label Color:	#FFFFFF
Label Font:	Arial
Label Size:	40

✓ Color Map

Interpolation: linear

Color	View	Quantity
#0000FF	🟦	4.5
#00FF00	🟩	5
#FF0000	🟥	5.5
#FFFF00	🟨	6
#e717e7	🟪	6.5
#45bab3	🟩	7
#653dc2	🟪	7.5

Method: TINMethod
Resolution: 100
Opacity: 155
MDS URL: http://karlinapp.ethz.ch/fcgi-bin/qgis_map_server/

Lat:53.666192¦,Lon:20.494981¦

Figure 42. The SANY Decision Support System [copyright SANY Consortium]

The SANY DSS Web mapping component uses the GetMap as its main operation for the visualization of the fusion results. The graphical user interface of the SANY DSS lets the user parameterize an appropriate style (e.g. choropleths and contours), then assembles the request for a corresponding GetMap operation (which includes the user style) and finally it performs the request in order to obtain the desired visualization in real-time. In addition, spatial navigation (zoom, pan, recenter) and layer management is always controlled by the parameters of the GetMap request message, so that the user always receives the appropriate visualization within the interactive map.

7.1.4 Highlighting the Advantages of Service-driven Cartography for Up-to-Date and Interactive Maps

As discussed in chapter 5, in service-driven cartography the implementation of cartographic web services is mediating between the data and the cartographic application. Furthermore, based on the map representation, it automatically handles all steps necessary for transforming the geodata into a map, while offering a clear interface to the client applications for accessing and displaying the generated maps. As a consequence, the SANY DSS is providing always a visualization of information that is always up-to-date, which is an important requirement for decision support systems.

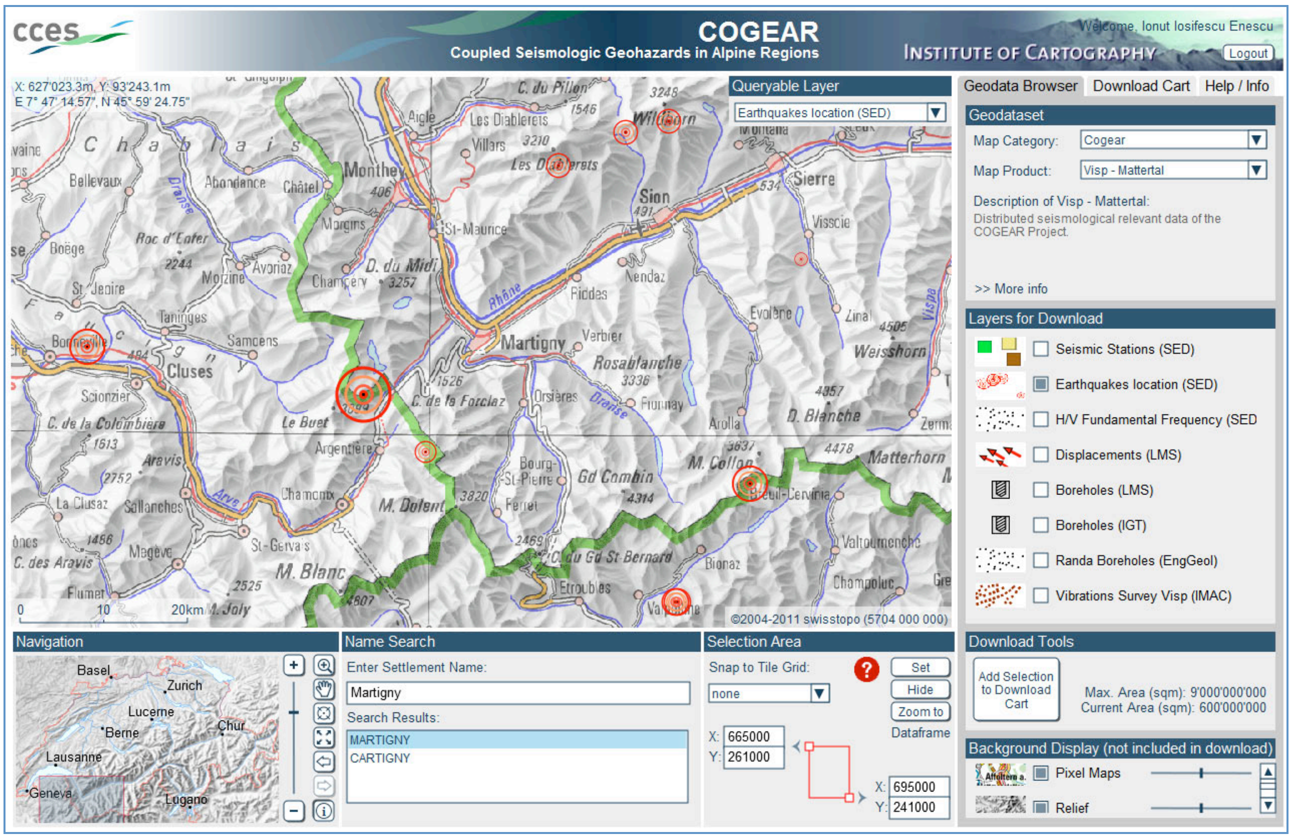
Furthermore, since cartographic web services are mainly designed to interpret a given map representation in order to produce the map, a client application can change the map output by changing the map representation. The SANY DSS makes use of the functionalities for accessing and changing the representation of the maps presented in the map viewer, thus providing a high level of interactivity for customizing the map appearance as visible in the SLD Settings window from *Figure 42*.

7.2 Expressive and Reusable Map Representations

7.2.1 The GIS Platform for Environmental Research

The concept of service-driven cartography is further applied in a GIS Platform for Multidisciplinary Environmental Research developed for the COGEAR and SwissExperiment projects [Iosifescu et al., 2010d]. The COGEAR project is an interdisciplinary natural hazard project for investigating the hazard chain induced by earthquakes. It addresses tectonic processes and the related variability of seismicity in space and time, earthquake forecasting and short-term precursors, and strong ground motion. The SwissExperiment project is an initiative of the Competence Centre Environment and Sustainability of the ETH Domain (CCES), which has been created to provide a platform for large scale sensor network deployment and information retrieval and exploitation. Its main goal is to enable effective real-time environmental monitoring through wireless sensor networks and a common, modern, generic cyber-infrastructure. This infrastructure will be used to work efficiently and collaboratively in finding the key mechanisms in the triggering of natural hazards and efficiently distribute the information to increase public awareness [Iosifescu et al., 2010d].

The GIS Platform for Multidisciplinary Environmental Research is intended to be part of the modern infrastructure for seismic and more general natural hazards research. It organizes research data, sensor information and additional large topographic datasets needed for environmental research in one coherent platform as shown in *Figure 43*.



Map Category: Cogear
Map Product: Visp - Mattertal
Description of Visp - Mattertal:
Distributed seismological relevant data of the COGEAR Project.
>> More info

Enter Settlement Name: Martigny
Search Results:
MARTIGNY
CARTIGNY

Snap to Tile Grid: none
X: 665000
Y: 261000
X: 695000
Y: 241000

Max. Area (sqm): 9'000'000'000
Current Area (sqm): 600'000'000

Figure 43. COGEAR GIS Platform prototype
basemap reproduced with the authorization of swisstopo (JA100120); basemap source: Bundesamt für Landestopographie swisstopo (Art. 30 GeoIV): 5704 000 000

7.2.2 The Design of the GIS Platform

Due to the extreme heterogeneous and distributed nature of the research data, the COGEAR data exchange was designed to abide the generic, distributed architecture presented in *Figure 44* [Iosifescu et al., 2010e].

The distributed GIS Platform Architecture has three components, namely a Data Management Component, a Computation and an Interaction component, which correspond to the data tier, logic tier and presentation tier from the well-known three-tier architecture [Eckerson, 1995]. The data tier is represented by several distributed databases, the application tier is represented by services for visualization and download and the presentation tier is represented by the user interface [Iosifescu et al., 2010d].

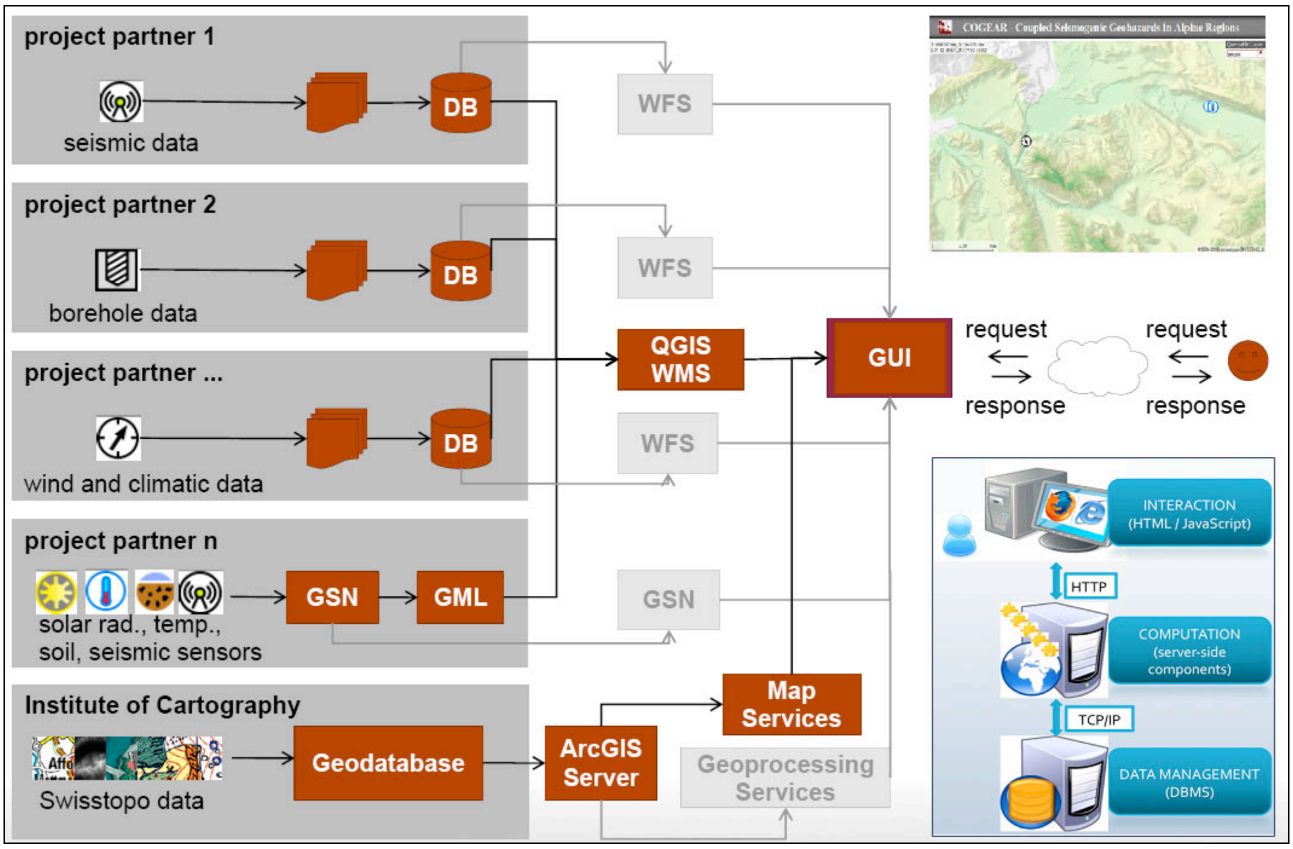


Figure 44. GIS Platform Architecture [Iosifescu et al., 2010e]

In this design, the cartographic web services do not longer play the central role, but rather their support for OGC standards and their integration with other services for data viewing, geoprocessing and downloading. Standards-based interoperability is a necessity when Cartographic Web Services (CartoWS) must cooperate with various other system components such as Web Map Services (WMS), Geocoding Services, Web Feature Services (WFS), Global Sensor Networks (GSN) and Geoprocessing Services. This cooperation is mandatory in order to build complex cartographic platforms such as the GIS Platform.

7.2.3 Cartographic Functionalities of the GIS Platform for Environmental Research

The main role of the GIS platform is to communicate ongoing investigations and their generated data to all potentially interested researchers. It visualizes the relevant research data in order to geographically identify the overlapping areas of interest and to enable research synergies. As shown in *Figure 45*, sensor information can be visualized on maps in real-time by using signatures that can be naturally understood by researchers thus allowing them to gain better insights and detect spatial patterns before further proceeding to the selection and download of the datasets.

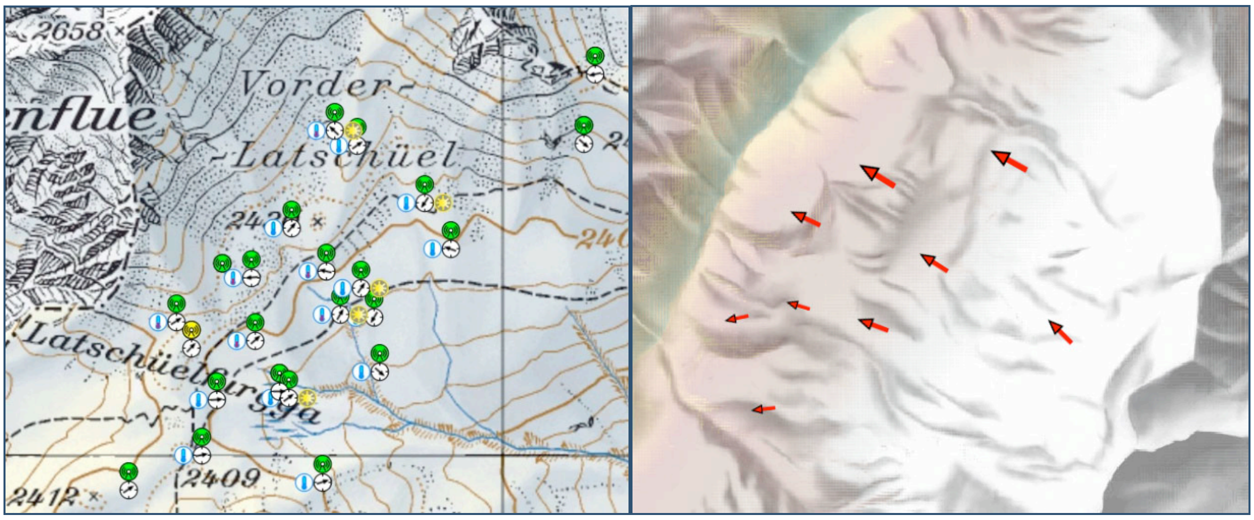


Figure 45. Example visualizations of research data in the GIS Platform [Iosifescu et al., 2010d] basemap reproduced with the authorization of swisstopo (JA100120); basemap source: Bundesamt für Landestopographie swisstopo (Art. 30 GeoIV): 5704 000 000

These cartographic functionalities will be further enhanced with cartographic tools for visual data mining. The cartographic tools will improve the possibilities to derive new information by offering similarly as in the SANY DSS, a visualization of interpolated sensor point measurements and by allowing users to customize class boundaries and set thresholds interactively.

7.2.4 Highlighting the Advantages of Service-driven Cartography for Expressive and Reusable Map Representations

The map representation shows its potential for environmental monitoring and decision support as it makes use of several specific features discussed in chapter 4 such as SVG symbols and patterns.

Sensor information such as temperature, wind speed, wind direction, solar radiance is represented on maps in real-time using map representations. The GIS platform can be currently used for the monitoring of sensor status (green – the sensor is delivering data, yellow – the sensor has errors in the data provision) and for the monitoring of the streamed data. Sensor measurements are related with graphic variables such as color, shape, size and rotation of the symbols in the map representation as it can be observed in *Figure 45*.

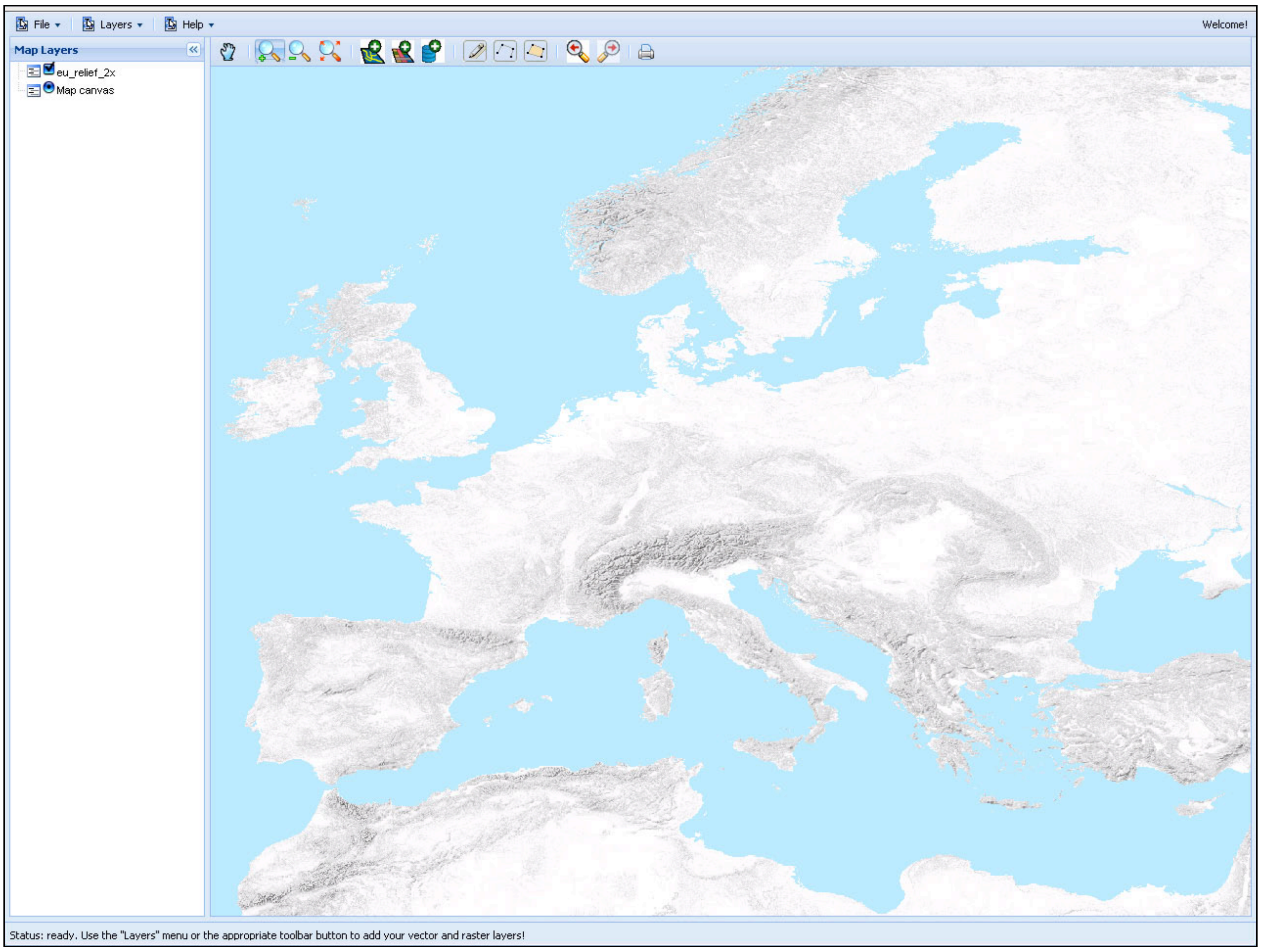
Furthermore the COGEAR GIS platform not only proves the use of on-demand mapping for environmental data as in the case of the SANY DSS, but it also demonstrates the flexibility, expressivity and reusability of map representation for multiple data sources.

The map representations allow defining the desired map symbolization decoupled from the physical data therefore insuring that maps are created from the latest data available. Moreover, several layers and sensor measurements types can be overlaid in one map frame for an overview of the environmental parameters in the area. In many cases, the map representation is also reused for new data sources (e.g. boreholes information, displacements information, GSN sensor data), which are similar to layers that already exist in the platform.

7.3 Flexible and Interactive Cartographic Applications

7.3.1 Prototype WebGIS Software

The general flexibility of the specifications and software presented in this work is further demonstrated in regard with its support for interactive GIS software as shown in the next illustration. The software depicted in *Figure 46* represents a proof-of-concept WebGIS software [Iosifescu, 2009].



Status: ready. Use the "Layers" menu or the appropriate toolbar button to add your vector and raster layers!

Figure 46. Prototype WebGIS Software [Iosifescu, 2009]

It must be stressed that the system is not just a GIS application, but also much more: GIS software. This means that the WebGIS offers to the users the possibility to upload and symbolize their own data, online and without the need to install any GIS software on their workstations. This setup is possible because, on one hand, the map representation is separated from the GIS data, and on the other hand, the user interactions with the WebGIS regarding the symbolization of the data can be parameterized into the map representation. Further on, the map pane always shows the rendering of the data based on the latest map representation, therefore effectively enabling a Web-based interactive GIS.

7.3.2 The Implementation Design of the WebGIS Prototype

The implementation design of the WebGIS prototype follows a three-tier architecture, as previously discussed in the COGEAR GIS Platform Architecture, with a Data Management Component, a Computation and an Interaction component. An overview of the implementation of the WebGIS prototype is presented in the following overview figure:

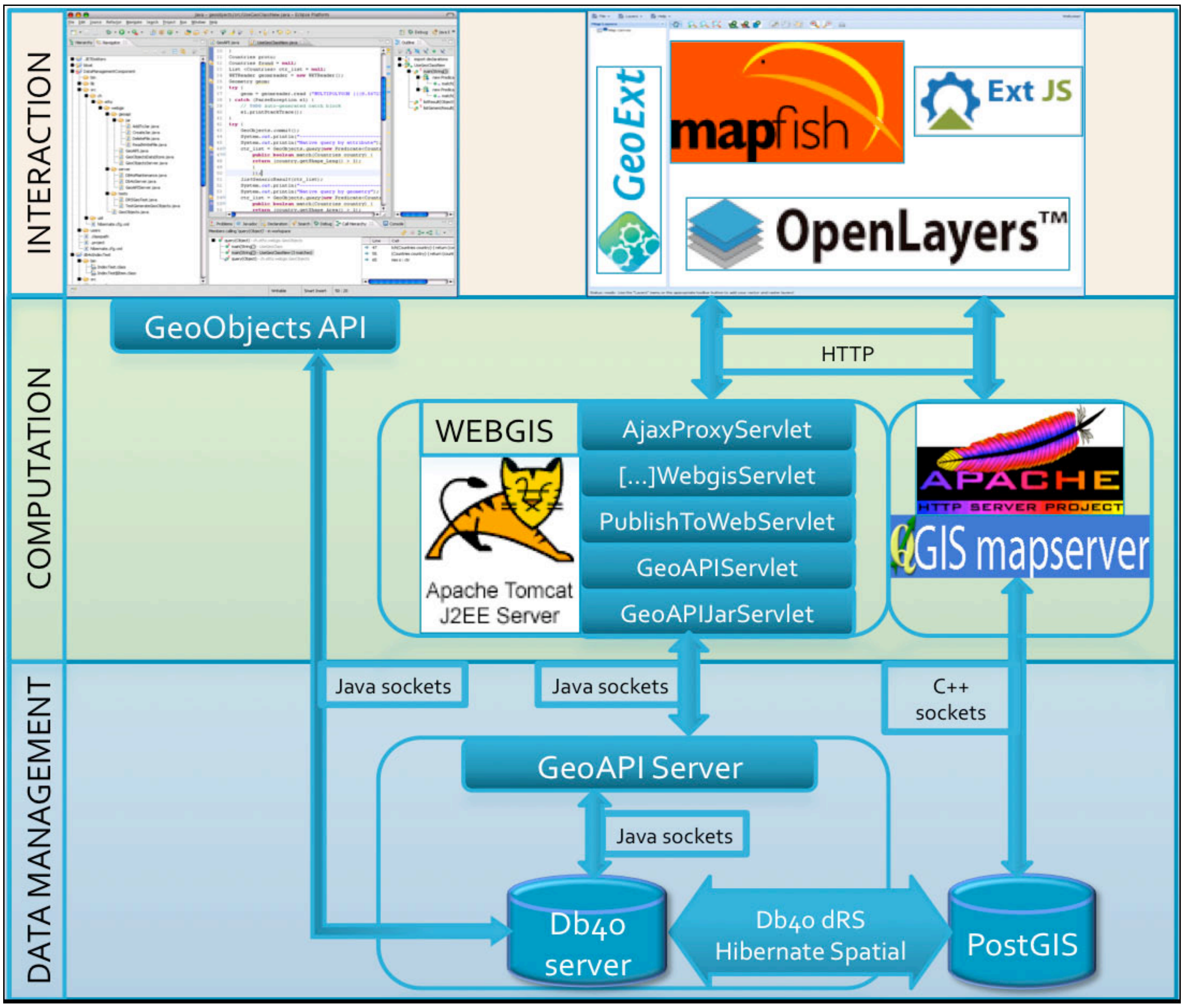


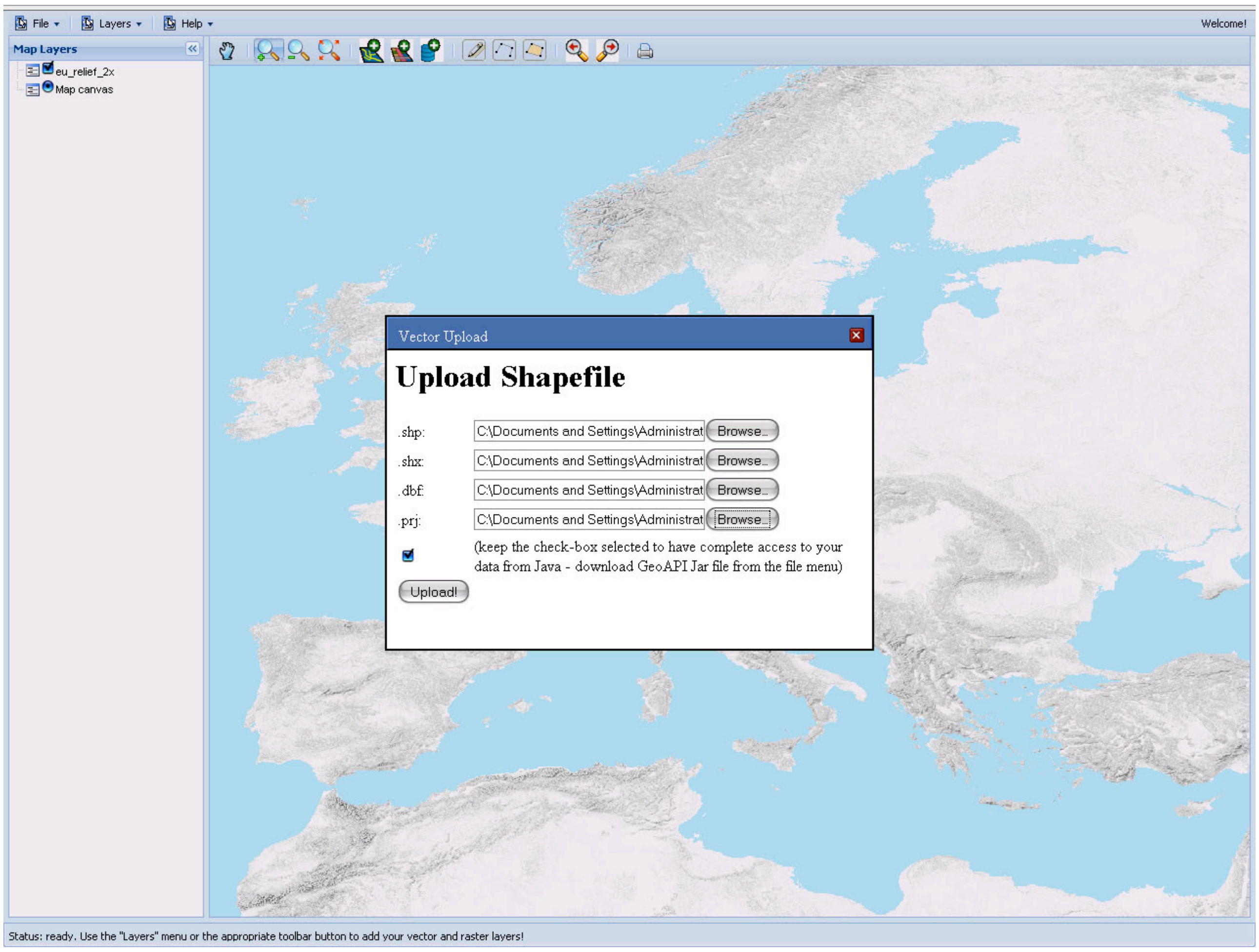
Figure 47. The implementation design of the WebGIS prototype [Iosifescu, 2009]

The data management component was developed around databases, the computation component was implemented using JavaServlets/JSP and the QGIS mapserver, and the user interaction was mainly implemented based on existing web mapping frameworks such as MapFish [Iosifescu, 2009]. Cartographic Web Services are at the core of the map visualization system through the use of the QGIS mapserver.

7.3.3 Interactive Functionalities of the Web GIS

The functionalities implemented in the WebGIS are less related to cartographic expressiveness but rather to the interactive possibilities for remote visualization and authoring of maps from GIS data via Web-based software.

The first functionality allows for the upload of user data in raster and vector formats as illustrated in *Figure 48*.



Vector Upload

Upload Shapefile

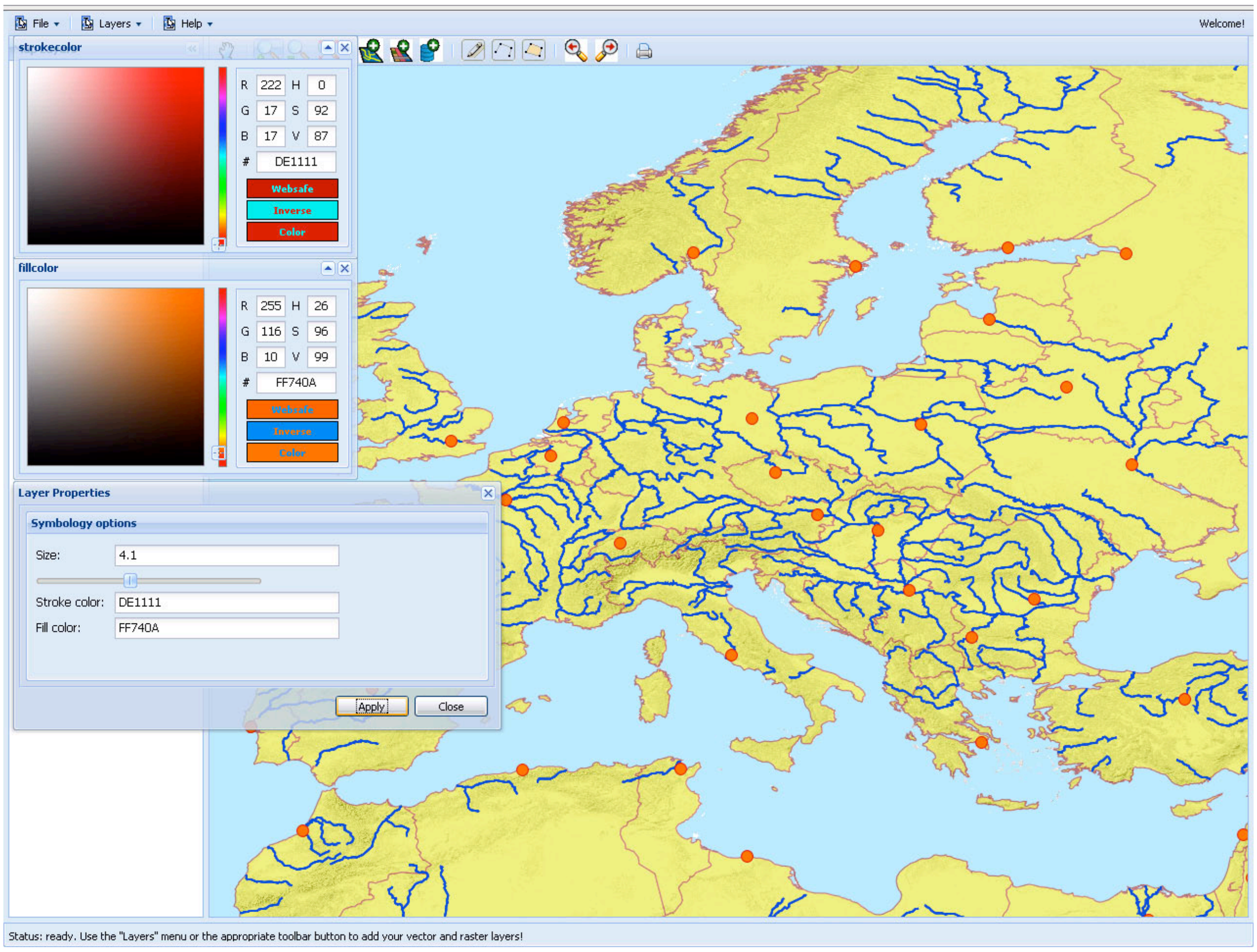
.shp:	C:\Documents and Settings\Administrat
.shx:	C:\Documents and Settings\Administrat
.dbf:	C:\Documents and Settings\Administrat
.prj:	C:\Documents and Settings\Administrat

✓ (keep the check-box selected to have complete access to your data from Java - download GeoAPI Jar file from the file menu)

Status: ready. Use the "Layers" menu or the appropriate toolbar button to add your vector and raster layers!

Figure 48. WebGIS functionality for upload of GIS data [Iosifescu, 2009]

At the end of the uploading process, the new dataset will be added to the map and listed in the layer tree of the user interface. From the contextual menu of the new layer a user can access the panels for the configuration of the basic symbology. In this manner, a user can upload and interactively symbolize multiple GIS data sets as shown in *Figure 49*.



strokecolor

R	222	H	0
G	17	S	92
B	17	V	87
#	DE1111		

fillcolor

R	255	H	26
G	116	S	96
B	10	V	99
#	FF740A		

Layer Properties

Symbology options

Size: 4.1

Stroke color: DE1111

Fill color: FF740A

Status: ready. Use the "Layers" menu or the appropriate toolbar button to add your vector and raster layers!

Figure 49. Map composition with several layers [Iosifescu, 2009]

The WebGIS has also additional interactive functionalities such as PDF map export and an adaptive interface for mobile devices as shown in *Figure 50*.

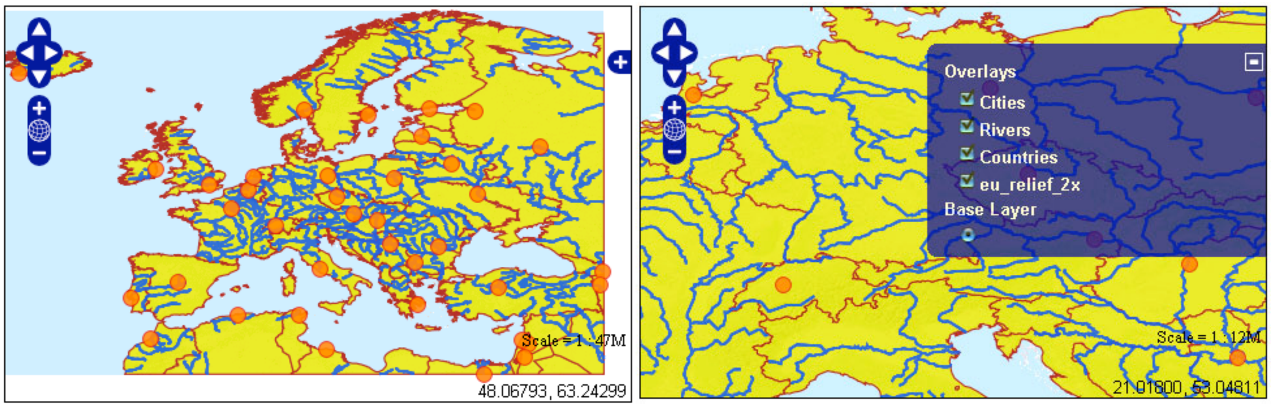


Figure 50. Mobile version of the interactive map [Iosifescu, 2009]

These functionalities are facilitated by the service-driven architecture. The export of the authored map as PDF (shown in *Figure 51*) is possible because the map images can be generated with a high DPI/resolution, a feature that supported by cartographic web services. Moreover, CartoWS allows to adapt the WebGIS software to multiple Web media. For example, the WebGIS mobile version is optimized for the reduced screen size of mobile devices such as iPhone and iPod Touch. The mobile version has been implemented without significant additional effort, especially because of the decoupling of the CartoWS generated map from the overall user interface.

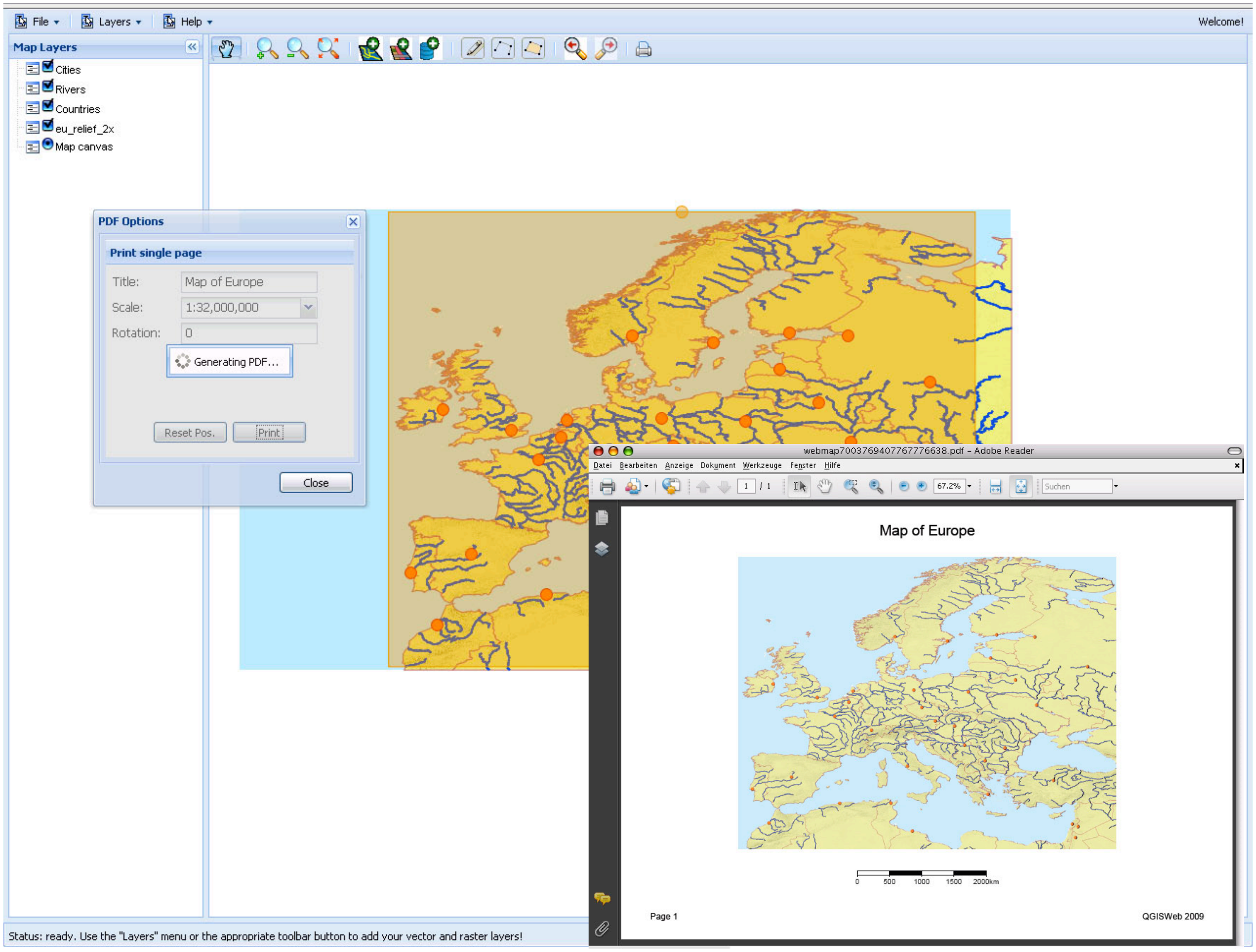


Figure 51. PDF map export [after Iosifescu, 2009]

7.3.4 Highlighting the Advantages of Service-driven Cartography for Flexible and Interactive Cartographic Systems

With the features presented so far, the WebGIS fulfills the basic requirements of a GIS user, namely viewing the spatial data, map production and online publishing of the map. As these functionalities are offered directly on the Web, the implemented WebGIS demonstrates the viability of map authoring in real time based on map representations, which are defined interactively within the WebGIS software.

Every interaction in the user interface triggers an event, which is bound to a JavaScript function. The JavaScript function will make an XMLHttpRequest (via the HTTP protocol) for a GetMap operation of QGIS mapserver. The request will always contain the map representation that must be applied to the data in the map pane of the user interface. The QGIS mapserver will then perform the task of rendering the spatial data into the current map image that is shown in the WebGIS. More details about the WebGIS and its additional functionalities can be found in [Iosifescu, 2009].

In conclusion, the successful implementation of SANY DSS, the GIS Platform for Environmental Research demonstrates the viability of CartoWS for practical cartographic applications. Moreover, the WebGIS proves the general applicability of CartoWS beyond the realm of environmental management and towards innovative Web cartographic applications that take advantage of the flexibility provided by service driven cartography.

8 Additional Research

This chapter contains additional research that was undertaken in close relation with this thesis. The problem statement of this thesis generated several intricate ramifications. The content in this chapter presents an overview on additional research directions that arose from the development of CartoWS and from the emerging area of service driven cartography.

8.1 Towards Generic Semi-Automatic Symbolization

The definition of CartoWS achieved the goals stated for this work. The “Publish to Web” plugin already improved the user experience by generating the structure of the map representation based on the map that is interactively authored in a graphical user interface. It alleviates much manual work by exporting the basic symbology in XML. For the map representations to work well, the data needs to possess attributes that can be used to control rotation, scale, size or even transparency of the individual features. For example, for an optimal display of labels, the dataset should preferably contain additional attributes that can be used for specifying the placement or displacement information for textual labels. However, such attributes cannot be defined for all cases, and it may be required that additional attributes are computed on demand each time a new map request is made. In such cases, cartographic algorithms are required to intelligently adapt the representation of the data to specific map scale and bounding box.

8.1.1 Towards Interoperable Algorithms

Such algorithmic complexity is beyond the expressive power provided by map representations expressed in SLD. In this context, a possible solution could be defined based on interoperable algorithms. The idea of using an interpreted programming language (or scripting) is introduced as backend for the standardization of cartographic algorithms and functions that go beyond the expressivity of the map representation. If defined, the underlying syntax set by a common CartoWS Application Programming Interface (API) would be accepted by all implementations of CartoWS. Based on a strictly defined syntax, algorithms could be expressed in this interpreted programming language. In this manner advanced cartographic algorithms are becoming scripts that can be exchanged, interpreted and executed by CartoWS. In theory, they can be exchanged without restrictions as long as they abide the common “API”. The main role of the interoperable algorithms is to further enhance the automatic map symbolization in situations that need an adaptive map representation such as multi-scale symbolization or label placement. For example, when scale-dependent information is not present in the source GIS data, the algorithms could automatically expand the map representation to handle the symbolization at various scales.

A basic proof of this concept for the further automation of map production was tested with a basic algorithm for label placement depending on the map scale and view extent. The test was implemented using the Python scripting language and the API of the QGIS Desktop. However, no particular language is favored for expressing such cartographic algorithms. Further results in this direction can be obtained through the analysis, extraction, harmonization and implementation of a generic API starting from existing GIS and graphic software APIs.

8.1.2 Automatic Symbolization based on Cartographic Conventions

The automatic generation of the map representation could be improved even further when we consider that many maps contain layers with symbology constrained by cartographic conventions. Cartographic conventions are derived from the traditions of map making. They are related to map quality in the sense that deviations from established norms may result in a map that is less effective at communicating the intended message. Color conventions are for example that forests and vegetative cover are green; hydrological features (rivers, lakes, oceans and other water bodies) are blue; the highways and principal roads are usually red, secondary roads are yellow, and less important roads are black; contour lines are brown; the color sequence of dark green, light green, yellow, orange, red and brown for increasing elevations; black represents the man-made features; etc. For example a small black triangle normally means a mountain summit. Other conventions recommend the use of pictorial symbols to attempt to mimic the real world. Such conventional symbols could be already available in symbol libraries. Conventions for linear features state that roads are usually solid or dashed lines, railways are hatched, and trails are often dotted lines, and boundaries could have stroke patterns such as dash-space-dash, or dash-space-dot-space-dash, etc. In addition, generic algorithms such as the ones for label placement could be used in conjunction with a TextSymbolizer whenever labeling is required.

The main idea would be to recognize the data, which is subject to cartographic conventions and let the software suggest a corresponding map representation for these specific features. In the best case, the suggested map representation can be accepted, thus significantly decreasing the efforts involved in defining a custom map representation. In the worst case, the rejection or further modification of the suggested symbology would not significantly increase the effort.

8.1.3 Towards Cartographic Ontologies

This work also researched the use of ontologies in order to set a solid foundation for the generic API in CartoWS and to enable the software to suggest the symbology for layers susceptible to cartographic conventions [Iosifescu et al., 2007]. In the context of computer science, ontology may be thought of as a formal representation of the knowledge associated with a particular domain, task, or application whose ultimate purpose is to enable machine understanding [Gruber, 1993].

Cartographic ontologies are intended to provide a source of predefined cartographic concepts for use within software that automatically creates the map representation. Cartographic ontologies could mediate between different computer programs, thus enabling them to share a common view on cartographic information and making map representations explicit and reusable. At the same time, they would allow to separate the domain knowledge such as the common cartographic vocabulary from the operational knowledge such as algorithms.

From a practical point of view, the cartographic ontologies can be embedded in the architecture of a cartographic library supporting the advanced cartographic processes for defining algorithms and suggestions for symbolization. The research in the possible usage of cartographic ontologies handled the remaining concepts of service-driven cartography that are beyond the syntactical definitions of map representations and CartoWS. With the definition of cartographic ontologies and their subsequent implementation in the cartographic library, the map representations could receive the syntax expressing complex cartographic algorithms as mentioned previously, effectively creating an “universal interpreted language” for cartographic applications.

Furthermore, we discovered that the automatic suggestion of various conventional map representations by a cartographic library requires spatial ontologies in addition to cartographic ontologies. Spatial ontologies are concerned with defining the semantics of the spatial features (what they are) while cartographic ontologies are concerned with cartographic representation of spatial features (how they can be represented). Cartographic ontologies and spatial ontologies are two different concepts. However, cartographic ontologies must enhance the use of spatial ontologies, because mapping software systems would need to use them both in order to assist in creating an appropriate symbolization.

The fundamental proof of these concepts was attempted based on the implementation of the label placement algorithm discussed in chapter 8.1.1. The Python script was enhanced to use a test cartographic ontology that was defined using the ontology editor Protégé [Protégé, 2010]. The ontology was then exported in the Web Ontology Language (OWL) format. As a consequence, it was not only possible to parse the ontology directly in the Python script, OWL being an XML-based format, but it also opened the possibility to link the FaCT++ reasoner [FaCT++, 2010] with the C++ libraries of the QGIS Desktop and later with the QGIS mapserver. The ontology contained generic cartographic concepts and a specific vocabulary based on the core API of the QGIS Desktop, the same API used by QGIS mapserver [Iosifescu et al., 2007]. The final step was the implementation of the labeling algorithm based on the vocabulary fixed by the ontology and using the Python scripting language and the Python bindings for QGIS. Even though this approach has the advantage of demonstrating the concept based ontology and algorithms, it is not generic in the sense that it is not universally applicable to any GIS datasets. First of all, without spatial ontologies (or at least complex metadata) it is impossible to make any software to recognize what the dataset is. Secondly, a generic API (extracted from existing GIS and graphic software APIs) for scripting cartographic algorithms must be completely defined. Finally, the implementation used only a small subset of the possible functionalities available in the QGIS software API, and the development and practical use of cartographic ontologies must be preceded by software for processing spatial ontologies and the generic algorithms for spatial processing and cartographic visualization.

As it can be noted from this subchapter, the research in CartoWS was enhanced with the inclusion of algorithms and cartographic conventions. However, the related efforts gradually became broader, therefore requiring further research in several neighboring but nevertheless complex topics such as in spatial ontologies and APIs for GIS. As consequence, future research can exceed the focus of CartoWS concepts. Nevertheless, these efforts show the intrinsic connections between service-driven cartography and related topics in GIS and computer science.

8.2 Towards Object-Oriented Symbology in GIS Data

The GIS features are commonly managed within a relational geodatabase model, which enforces the notion of a layer for geographic data. A layer is a set of geographic data with a common type or theme as for example cities, rivers, streets or buildings. In the relational model all spatial features of the same category are managed together within the same schema/table, and each record in the table corresponds to a spatial feature.

8.2.1 Filtering vs. Object-Oriented Symbology

Currently, the map representation requires heavy use of filter conditions in order to achieve fine-grained selection and symbolization of individual geometric features. This is evident in the lengthy Filter definition that must be created in order to assign a specific symbology to an individual feature or to a sparse group of features. Filtering works top down, starting with all features in a layer, and gradually eliminating the features that do not fit the defined selection predicates until only the desired feature remains. While this approach is very suitable for the assignment of symbology to entire layers, it is not well suited for individual features because of the significant effort required for defining the predicates for the selection. As a consequence, the map representation for CartoWS must contain relatively complex filter expressions that logically combine constraints on the properties of a feature in order to identify a particular subset of features to be operated upon.

This filtering complexity is actually due to the restriction of the relational model of storing and managing GIS data. Even though the GIS spatial data model is apparently not directly related to cartography, it influenced and restricted the research strategy for defining the map representation and the CartoWS since most of GIS software are based on the relational model and therefore manages the geodata accordingly.

The map representation and the CartoWS achieve the research goals for the current relational geospatial data model. However, towards the end of this work we have considered what would happen if an object-oriented (OO) model for spatial data would replace the current relational model and become mainstream.

Within an object-oriented approach, filtering and selection becomes unnecessary on a layer level therefore simplifying the global map representation through splitting in many individual and simpler representations. Individual spatial objects could then have one or more specific individual representation assigned to them as part of their behavior. In this manner spatial objects will know how to draw themselves on a canvas either in a GIS Graphical User Interface or in a map created by CartoWS. In summary, spatial objects would carry their own map representation and therefore the complexity of each individual map representation is greatly reduced. This approach can be very well suited for both the assignment of symbology to individual spatial features (local) as well as for entire layers (global). Simplified, this approach can work bottom up like a puzzle in order to compose the global map representation for the entire layer. As individual spatial objects have their own symbology, a layer would simply consist of a collection of spatial objects of the same base class. In the rendering of the final layer, each spatial object would contribute with its rendering behavior based on the individual map representation.

8.2.2 Object Oriented Models for Spatial Data

The fundamental place to start this vision is to have an object-oriented model instead of a relational model for spatial data, thus objects instead of relational features. This is conceivable, since spatial information was designed to natively support the conceptual modeling and storage of geographic features as objects. This association between the georeferenced geometry of a geographic feature and its attributes is the most important advancement that GIS introduced in comparison to digital cartographic systems or CAD (Computer-aided Design) systems. Distinct objects such as point, line or polygon features are stored, on one hand, with the coordinates of the features (geometry) and, on the other hand, with one or more attributes. Examples of such spatial objects are cities (as point entities), rivers (as linear entities) and country boundaries (as polygon entities). Moreover, the GIS object oriented conceptual model has the advantage of encapsulating in each geographic entity, not only its properties but also, in theory, methods for a domain specific handling of this data such as for rendering itself on the canvas or setting, updating and returning the associated map representation.

Although the object oriented (OO) approach is a natural fit for spatial data, in existing systems, vector data is represented as relations. Vector data is stored in various file formats or, more usual, in spatially enabled database tables. In relational tables, each tuple contains the geometry and the additional attributes of the individual objects in space, which are usually assigned a feature id in order to be identified. In the relational model, all spatial objects of the same category are managed within the same schema/table. Therefore, real world objects are flattened in a table record, with no possibility to define or store their behavior through methods that would be an inherent part of the information contained by the spatial object.

Additional research was therefore performed on the fundamental level of obtaining spatial objects as a basis for object-oriented symbolization. For the transition from relation to object oriented handling of GIS data, it is necessary to be able to automatically import and use existing relational data sources in the application. Therefore a mechanism was defined in order to transform existing spatial relational tables in class definitions [Iosifescu, 2009]. It was found that such an automatic transformation is possible because the relational model is less expressive than the object oriented model, since relations have no behaviors linked to individual records. As consequence, domain classes can be derived directly based on the relational model of the data and these domain classes can be later enhanced with methods that define their behavior.

The object-oriented enablement of spatial data poses additional challenges regarding its storage and life cycle, which required further research in the domain of spatially enabled Object Oriented Database Management Systems (OODBMS). These considerations gradually exceed the focus of the research on Cartographic Web Services and the intermediary results that were presented show only the need for further research.

9 Conclusions and Outlook

This chapter contains a summary of the thesis's results, as well as conclusions and future work. It begins with a synopsis of results, most notably the definition of a standards-based map representation for cartography, the specification of Cartographic Web Services and their implementation in software. The synopsis of results continues with the main conclusions of the thesis, namely the map representation as a means for formal documentation of maps for posterity and the use of Cartographic Web Services as a future-oriented workflow for automatic generation of maps from known GIS data. The chapter ends with an outlook on further research and the future perspectives of service driven cartography.

9.1 Synopsis of Results

This work advances the field of Web Cartography by defining what cartographic web services are and how can they are useful for the development of Web cartographic applications. It introduces cartographic Web services (CartoWS) as a solution for automatically producing maps from geodata based on a generic and expressive map representation. The maps are obtained from GIS data as geo-referenced images via a Web-based interface.

The core research started with an analysis of state of the art for a map representation for cartography. The gained knowledge helped in shaping a map representation based on the open SLD/SE OGC standards that are enhanced with cartographic requirements for the symbolization of topographic and thematic data. The specification of the map representation for cartography formally defines the syntax for expressing the map symbolization with a high degree of detail. Moreover, the map representation includes operators for the manipulation of geometries based on GIS concepts.

The concept of CartoWS is specified by the Map and Diagram Service Interface (MDSI). Making use of specific computer science knowledge, such as Web services and their mechanism, concepts and properties, we specify the interface that details the operations of CartoWS. The definition of the MDSI follows and enhances the Web Map Service (WMS) standard. The QGIS map server is the software implementation of the MDSI. Furthermore, the QGIS map server demonstrates the utility of CartoWS by several proof-of-concept applications that highlight the various aspects of service-driven cartography. Overall, an important advancement of this work in the realm of Web cartography is the innovative service-driven workflow for map-making, which is enabled by the resulted CartoWS specifications and software. The results of this work can be applied for the development of Web cartographic applications.

9.2 Conclusions

Automatic generation of maps from known GIS data is possible and highly practical. The syntactic definition of the open and generic map representation and its implementation in a Cartographic Web Service proved the hypothesis stated in the beginning of this thesis: once the map is defined within a map representation, the Cartographic Web Service automatically selects, symbolizes and ultimately renders the map in real-time and without any user intervention.

The defined map representation is generic and expressive and offers the means for the formalization of maps. The formal description of map content via the map representation fosters awareness to cartographic science. It also preserves cartographer’s choices and it allows future generations to follow and understand how a certain map was created.

The use of CartoWS and service driven cartography is aimed at two types of users: cartographers and GIS specialists. On one side, service driven cartography enables cartographers to produce interactive maps for the Web directly from GIS data with little effort and based on formal cartographic representations. On the other side, GIS specialists can use CartoWS to improve the quality of their cartographic output in their Web GIS applications as well as to exchange and reuse the open map representation (styling) in other projects.

To visualize the spatial and thematic data based solely on a descriptive map representation is an important change in the map production workflow. The definition of a map representation is more challenging than the manual symbolization in a graphic program, but its usefulness becomes visible mostly due to the enforced separation between symbology and the data itself. Not only the map representation provides the possibility to define the symbolization of the current state of the data, it can also handle the progression of time for the updated versions of the data.

Interoperability and reusability are of great concern in GIS. Therefore, the CartoWS was defined based on well-known and open standards. Besides, the use of GIS standards reduces the technological gap between GIS and cartography by insuring the interoperability and integration of service-driven cartography with the state-of-the art developments from the GIS side.

With the current trend towards Web-based applications for daily computing tasks, the Cartographic Web Services provide a future-oriented foundation of map-making from GIS data. This contribution can be seen as a logical step and innovation towards a closer cooperation between GIS and Cartography for mutual beneficial influence.

9.3 Outlook

Cartographic Web Services were designed to provide a solution to the fundamental problems stated at the beginning of this thesis, but the future research can expand the map representation to handle advanced cartographic issues such as automatic generalization, advanced relief shading, automatic cliff drawing, automatic label placement or advanced line symbolization.

Automatic generalization for topographic maps is a very complex process using principles [SGK, 2005] that can be difficult to apply and implement in software for every single situation. There are various generalization operators as simplification of geometry and elimination of unimportant elements that can be used as building blocks for generalization algorithms for vector data. Raster data can be generalized in a similar manner by constructing pyramids and tiles for different levels of detail. Currently, the use of CartoWS starts with the premise that multi scale spatial repositories are already available, and they were created either using manual generalization or certain generalization algorithms. However, there are already a multitude of generalization methods, and future research may attempt to couple CartoWS with the realm of automatic generalization.

In chapter 4, we have seen that a map representation can contain the definition of shaded relief. However, the definition does not mention anything about the algorithm to be used for the rendering of the relief. Future research may pursue to improve on service-driven relief shading as a method for representing geomorphology on maps in a natural, aesthetic, and intuitive manner [Imhof, 1982; Jenny et al., 2006]. In addition, cliff drawing as a method for improving the clarity and aesthetic of cliff area in a map is a very good example for the artistic side of cartography. Automatic cliff drawing it is also possible, but usually cliff drawing is a very specific case of texturing that for best results is normally drawn manually. Future research can attempt to merge existing techniques in digital cliff drawing [Hurni, 1995; Dahinden, 2007] within Cartographic Web Services.

Finally, the research in cartographic line drawing tries to define novel drawing algorithms and texture mapping for 2D lines, another advanced feature in computer-assisted cartography [He, 2007]. The implementation of such complex algorithms for line drawing inside the CartoWS may further improve the cartographic rendering quality of the generated maps.

Future research may also address improving the syntax and usage of the current map representation. In this context, the spatial operators still have some limitations. For example, a limitation of the buffer spatial operator is its symmetry. Asymmetric buffers could be used to create more complex geometries and thus to achieve more complex cartographic symbolization of features. However, the introduction of complex and more exotic GIS processing features such as asymmetric buffers was not performed in order not to over-increase the complexity of the map representation. Asymmetric symbolization can be achieved nevertheless with symbol displacements and thus it can be performed on the symbolization level.

The advanced use of the proposed geometry spatial operators not only provides cartographers with the expressiveness and flexibility required for the creation of map representations but it may play an additional role: since every layer has its own map representation, Geometry Spatial Operators might be used for the detection and elimination of cartographic conflicts in the composition of new maps based on the recombination of layers from existing map representations.

Furthermore, the advancements in service-driven cartography presented in this thesis open new research questions in neighboring research domains, which gradually expand to reveal additional fundamental issues. For example, the availability of the basic concepts opened the view for further research in semi-automatic generation of the map representation through the definition and use of cartographic and spatial ontologies. Moreover, the map representation could be simplified with the concept of smaller cooperating components through the use of object-oriented management for the spatial features. The future research hints that the broader use of service driven cartography may trigger the emergence of new research directions in both GIS and Cartography.

Finally, the appropriateness of service-driven cartography for professional Web Atlases must be investigated further. With the orientation towards web access, the next major step would be to create atlases using a service-oriented architecture. The cornerstone for future developments is to evaluate the appropriateness of service-driven cartography for the next generation of Web-based thematic atlases, which must provide comparable map quality with existing desktop-based atlases, while increasing the effectiveness and decreasing costs of editorial work.

List of Figures

Glossary and Abbreviations

AIS	Atlas Information System
API	Application Programming Interface
CAD	Computer-Aided Design
CartoML	Cartographic Markup Language
CartoWS	Cartographic Web Services
CGI	Common Gateway Interface
CRS	Coordinate Reference System
CSS	Cascading Style Sheets
DBMS	Database Management System; examples are PostgreSQL, DB2, ORACLE
DEM	Digital Elevation Model
DiaML	Diagram Markup Language
DSS	Decision Support System
ESRI	Environmental Systems Research Institute
FES	Filter Encoding Standard
GEOWARN	Geospatial Warning Systems, Project
GIS	Geographic Information System
GML	Geography Markup Language
GUI	Graphical User Interface
HTML	HyperText Markup Language
IBM DB2	A commercial relational DBMS from International Business Machines
ICA	International Cartographic Association
IEEE	Institute of Electrical and Electronics Engineers

ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
OO	Object-Oriented
OODBMS	Object Oriented Database Management Systems
ORACLE	A commercial relational DBMS.
ORCHESTRA	Open Architecture and Spatial Data Infrastructure for Risk Management, Project
PEUNHA	Pan-European assessment of Natural Hazards
PostgreSQL	An open source relational DBMS
QGIS	Quantum GIS, an user-friendly desktop GIS
REST	REpresentational State Transfer
SANY	European Integrated Project Sensors Anywhere, Project
SE	Symbology Encoding
SLD	Styled Layer Descriptor
SOA	Service Oriented Architectures
SOAP	Simple Object Access Protocol
SVG	Scalable Vector Graphics
UML	Unified Modeling Language
URI	Uniform Resource Identificator
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WFS	Web Feature Service
WMS	Web Map Service
WSDL	Web Services Description Language
WWW	World Wide Web
XML	eXtensible Markup Language

References

Alonso, G., Casati, F., Kuno, H. and Machiraju, V. (2004). Web Services. Concepts, Architectures and Applications. Springer, Berlin, p. 354.

Booth, D. and Cayang, K. (2007) (Eds.). Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. http://www.w3.org/TR/wsdl20-primer/, accessed 2007-11-05.

Buehler, K. and McKee, L. (1998). The OpenGIS Guide. Open GIS Consortium Technical Commitee, Wayland.

Burrough P. A. and McDonnell R. A. (1998). Principles of Geographical Information Systems. New York, Oxford University Press.

Cammack, R. (2007). Cartographic Approaches to Web Mapping Services. In: Multimedia Cartography, 2nd edition, Cartwright, W., Peterson, M.P. and Gartner, G. (Eds.), Springer, pp. 441-453.

Cartwright, W. (1994) (Eds.). Interactive Multimedia for Mapping. Visualisation in Modern Cartography, MacEachren and Taylor, Elsevier Science, New York, p. 345.

Cartwright, W. (2007). Development of Multimedia. In: Multimedia Cartography, 2nd edition, Cartwright, W., Peterson, M.P. and Gartner, G. (Eds.), Springer, pp. 11–34.

Cerami, E. (2002). Web Services Essentials. Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, O'Reilly & Associates, Inc., Sebastopol, CA, p. 288.

Crosswell, P. (2000). The Role of Standards in support of GDI. In: Geospatial Data Infrastructures – Concepts, Cases und Good Practice, Groot, R. and McLaughlin, J. (Eds.), Oxford University Press, pp. 57-83.

Dahinden, T. (2008). Methoden und Beurteilungskriterien für die analytische Felsdarstellung in topografischen Karten. PhD Thesis, Institute of Cartography, ETH Zurich.

Dent, B. (1996). Cartography: Thematic map design. Wm. C. Brown Publishers.

Duschene P. and Sonnet J. (2004). WMS Change Request: Support for WSDL and SOAP, OGC Discussion Paper.

Dykes, J., MacEachren, A. and Kraak, M.-J. (2005). Exploring Geovisualisation. Elsevier.

Eberle, A. (2009). Visualisierung von Naturgefahrendaten mit QGIS MapServer und OpenLayers. BSc Thesis. ETH Zurich.

EC (2007). Directive 2007/2/EC of the European Parliament and of the Council establishing an Infrastructure for Spatial Information in the European Community (INSPIRE).

e-cartouche (2011). Cartography for Swiss Higher Education. Module Standardisation and Webservices (http://www.e-cartouche.ch), accessed 2010-10-10.

Erl, T. (2008). SOA: Principles of Service Design. Prentice Hall. ISBN 0-13-234482-3.

FaCT++ (2010). FaCT++ (http://owl.man.ac.uk/factplusplus/), accessed 2010-12-10.

Glennon, R.M., and Glennon, J.A. (2007). Making Better Maps with ArcGIS 9.2. ArcUser, Summer (July- September) 2007, pp. 59-62.

Gogu, R., Dietrich, V.; Jenny, B.; Schwandner, F. and Hurni L. (2005): A geo-spatial data management system for potentially active volcanoes. In Computers & Geosciences 32 (1), available through Science Direct: http://dx.doi.org/10.1016/j.cageo.2005.04.004

Green D. and Bossomaier T. (2002). Online GIS and Spatial Metadata. London and New York, Taylor & Francis.

Gronmo, R., Berre, A., Solheim, I., Hoff, H. and Lantz, K. (2000). DISGIS: An Interoperability Framework for GIS - Using the ISO/TC 211 Model-based Approach. http://heim.ifi.uio.no/~roygr/GSDI-2000.pdf, accessed 2010-12-10.

Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specification. Knowledge Acquisition 5, pp. 199-220.

Hake, G., Grünreich, D. and Meng, L. (2002). Kartographie. Walter de Gruyter, p. 604.

He, F. (2008). Novel drawing algorithms and application of texture mapping for 2D cartographic line symbolization. PhD Thesis, Institute of Cartography, ETH Zurich.

Hugentobler, M., Iosifescu, I. and Hurni, L. (2007). A Design Concept for Implementing Interoperable Cartographic Services based on reusable GIS Components. In Proceedings of the International Cartographic Conference, Moscow, Russia.

Hurni, L. (1995). Modellhafte Arbeitsabläufe zur digitalen Erstellung von topographischen und geologischen Karten und dreidimensionalen Visualisierungen. PhD Thesis, ETH Zurich.

Hurni, L. (2008). Multimedia Atlas Information Systems. In: Encyclopedia of GIS, Shekhar, S. and Xiong, H. (Eds.), Springer, pp. 759-763

Hurni, L. and Iosifescu, I. (2006a). What Makes Cartography Interesting for Mountain Researchers? In Proceedings of the 5th ICA Mountain Cartography Workshop, Bohinj, Slovenia.

Hurni, L., Gogu, R. and Iosifescu, I. (2006b). Cartography for Environmental Systems. Proceedings of the Workshop Intelligent Embedded Systems, July 20, 2006, Romanian Academy Library, Bucharest, pp. 9-17.

Hurni, L., Jenny, B., Terribilini, A., Freimark, H., Schwandner, F. M., Gogu, R. and Dietrich, V. J. (2004). GEOWARN: Ein Internet-basiertes Multimedia-Atlas-Informationssystem für vulkanologische Anwendungen. Kartographische Nachrichten 54 (2), pp. 67-72.

ICA (2010). ICA Commission on Maps and the Internet, http://maps.unomaha.edu/ica/Default.html, accessed 2010-12-10.

Imhof, E. (1982). Cartographic Relief Presentation. de Gruyter, Berlin.

Iosifescu, I. (2007a). Specification of the Map and Diagram Service. ORCHESTRA Consortium Deliverable, http://www.eu-orchestra.org/download.php?file=docs/OA-Specs/Map_and_Diagram_Service_Specification_v3.0-ETHZ.pdf, accessed 2010-12-15.

Iosifescu, I. (2007b). Implementation Specification of the OA Map and Diagram Service, ORCHESTRA Consortium Deliverable, http://www.eu-orchestra.org/download.php?file=docs/OA-Specs/Map_and_Diagram_Service_Specification_v3.0-ETHZ.pdf, accessed 2010-12-15.

Iosifescu, I. (2007c). OGC CR 07-105: SE Implementation Specification Change Request – extensions for thematic mapping. Open Geospatial Consortium

Iosifescu, I. (2009). Design of a General Architecture for GIS. MSc. Thesis ETH Zurich, Global Information Systems Group.

Iosifescu, I., Gogu, R., Neumann, A. and Hurni, L. (2006). Learning platform for GIS-based natural hazards mapping. In: Proceedings of the 5th European Congress on Regional Geoscientific Cartography and Information Systems (1), pp. 388-391, Barcelona.

Iosifescu, I. and Hurni, L. (2007). Towards Cartographic Ontologies or "How Computers Learn Cartography". In Proceedings of the International Cartographic Conference, Moscow, Russia.

Iosifescu, I., Hubentobler, M. and Hurni, L. (2009). From GIS & sensor data to Web maps (workshop). In Proceedings of the International Symposium on Environmental Software Systems (ISESS), Venice, Italy.

Iosifescu, I., Hubentobler, M. and Hurni, L. (2010a). Web cartography with open standards – A solution to cartographic challenges of environmental management. Environmental Modelling & Software 25 (9), pp. 988–999.

Iosifescu, I., Hubentobler, M. and Hurni, L. (2010b). Development of Web GIS and Web Mapping Applications with QGIS mapserver (tutorial reader). In Proceedings of the GIScience 2010 Conference, Zurich, Switzerland.

Iosifescu I. and Hurni L. (2010c). GIS Platform for Interdisciplinary Environmental Research, in Proceedings of the 7th ICA Mountain Cartography Workshop, Borsa, Romania (in press)

Iosifescu, I. and Hurni, L. (2010d). Web-service Driven Cartography, In: Proceedings of the Autocarto 2010 Conference, Orlando, Florida.

Iosifescu, I., Hurni, L., Fäh, D. and Dawes, N. (2010e): Poster: GIS Platform for Environmental Research, Latsis Symposium 2010, ETH Zurich.

ISO (2004). ISO 19125 - Geographic Information: Simple Features access (part1) and SQL option (part 2). International Organization for Standardization, ISO Technical Committee 211.

ISO (2005). ISO 19128 - Geographic Information: Web Map Server Interface. International Organization for Standardization, ISO Technical Committee 211.

Jaques, P. (2007). Decision Support Infrastructure Architectural Design. European Integrated Project SANY, SANY Consortium.

Jenny, B. and Hurni L. (2006). Swiss-style colour relief shading modulated by elevation and by exposure to illumination. The Cartographic Journal, 43-3, p. 198-207.

Jenny, B., Hutzler, E. and Hurni, L. (2010). Point pattern synthesis. The Cartographic Journal, 47-3, p. 257–261.

Klopfer, M. and Kannellopoulos, I. (2008) (Eds.). ORCHESTRA – An Open Service Architecture for Risk Management, ISBN 978-3-00-024284-7.

Klopfer, M. and Simonis, I. (2009) (Eds.). SANY – An Open Service Architecture for Sensor NEtworks, ISBN 9-783000-285714.

Leff, A. and Rayfield J. T. (2001). Web-application development using the model/view/controller design pattern. Proceedings of the Fifth IEEE International Enterprise Distributed Object Computing Conference EDOC-01, pp. 118-127.

Letho, L. (2003). A Standards-Based Architecture for Multi-purpose Publishing of Geodata on the Web. In: Peterson, M. (Ed.), Maps and the Internet. Amsterdam, Cambridge: Elsevier Press.

MacEachren, A. (1995). How Maps Work. Guilford Press, New York, London.

Mooney, P. and Winstanley, A., (2003). Mapping and Internet Based Public Transportation Journey Planning and Information Systems. In: Peterson, M. (Ed.), Maps and the Internet.

Amsterdam, Cambridge: Elsevier Press.

Neumann, A. (2005). Use of SVG and ECMAScript Technology for E-Learning Purposes, In Proceedings of the ISPRS Workshop 'Tools and Techniques for E-Learning', Potsdam, Germany.

Noy, N.F. and McGuinness, D.L. (2001). Ontology development 101: A guide to creating your first ontology. Stanford Medical Informatics, Stanford CA.

OGC (2000). Web Map Service Implementation Specifications, Open Geospatial Consortium.

OGC (2002a). Filter Encoding, Open Geospatial Consortium.

OGC (2002b). Geography Markup Language, Open Geospatial Consortium.

OGC (2002c). Styled Layer Descriptor Implementation Specification, Open Geospatial Consortium.

OGC (2002d). Web Feature Service, Open Geospatial Consortium.

OGC (2004). WMS Change Request: - Support for WSDL and SOAP, OGC 04-050r1, Open Geospatial Consortium.

OGC (2005a). Styled Layer Descriptor Profile of the Web Map Service Implementation Specification, Open Geospatial Consortium.

OGC (2005b). Symbology Encoding, Open Geospatial Consortium.

OGC (2007a). Press Release, http://www.opengeospatial.org/pressroom/pressreleases/761, accessed 2010-10-10

OGC (2007b). Reference Model for the ORCHESTRA Architecture Version 2. Editor: Usländer, T., OGC Best Practices Document 07-097.

OGC (2009). Filter Encoding 2.0 Encoding Standard, Open Geospatial Consortium.

OGC Standards Website (2010). http://www.opengeospatial.org/standards, accessed 2010-12-20

ORCHESTRA Book (2008). orchestra – an open service architecture for risk management. ISBN 978-3-00-024384-7.

ORCHESTRA Consortium (2007). Value of standards. http://www.eu-orchestra.org/TUs/Standards/en, accessed 2010-10-10.

ORCHESTRA Website (2010). http://www.eu-orchestra.org/documents.shtml, accessed 2010-12-20.

Ortner, F. (2011). Dienstbasierte Kartengenerierung für Web-Atlanten unter Anwendung erweiterter OGC-Standards. MSc. Thesis University of Zurich, Department of Geography co-supervised by ETH Zurich, Institute of Cartography and Geoinformation.

Patterson, T. and Hermann, M. (2010). Creating value-enhanced shaded relief in Photoshop. http://www.shadedrelief.com/value/value.html, accessed 2010-12-20.

Pautasso, C., Zimmerman, O. and Leymann, F. (2008). Restful Web Services vs. "Big"' Web Services: making the right architectural decision. http://www.jopera.org/files/www2008-restws-pautasso-zimmermann-leymann.pdf, accessed 2010-12-10.

Peterson, M.P. (2007). The Internet and multimedia cartography. In: Multimedia Cartography, 2nd edition, Cartwright, W., Peterson, M.P. and Gartner, G. (Eds.), Springer, pp. 35–50.

Protégé (2010). Protégé Ontology Editor. http://protege.stanford.edu/, accessed 2010-12-10.

QGIS (2010). QuantumGIS (http://qgis.org), accessed 2010-12-10.

QGIS mapserver (2010). Website of QGIS mapserver, http://karlinapp.ethz.ch/qgis_wms/index.html, accessed on 26.06.2010

Richardson, L. and Ruby, S. (2007). RESTful Web Services. O'Reilly, Sebastopol.

Richmond, E. and Keller, P. (2003). Internet Cartography and Official Tourism Destination Web Sites. In: Peterson, M. (Ed.), Maps and the Internet. Amsterdam, Cambridge: Elsevier Press.

Romer, S. (2010). Kartografische Funktionen in ArcGIS. MSc. Thesis ETH Zurich, Institute of Cartography.

Schnabel O. (2004) Map Symbol Brewer. In: Proceedings of the 3rd SVG Open Conference, Tokio, 2004, http://www.svgopen.org/2004, accessed 2010-12-10.

Schnabel, O. (2007). Benutzerdefinierte Diagrammsignaturen in Karten: Konzepte, Formalisierung und Implementationen. PhD Thesis, Institute of Cartography, ETH Zurich.

SGK (2005). SGK No. 17: Topographic Maps – Map Graphic and Generalization. http://www.kartographie.ch/publikationen/pdf/demo_no16_en.pdf, accessed 2010-12-22.

Sieber, R., Hollenstein, L., Odden, B. and Hurni, L. (2011). From Classic Atlas Design to Collaborative Platforms – The SwissAtlasPlatform Project. In Proceedings of the International Cartographic Conference, Paris, France (accepted for publication).

Slocum, T., McMaster R., Kessler, F. and Howard, H. (2005). Thematic Cartography and Geographic Visualization, second Ed., Pearson Education Inc., p. 518.

Sykora, P., Schnabel, O. and Iosifescu Enescu, I. (2007). Extended Cartographic Interfaces for Open Distributed Processing. Cartographica, Volume 42, Number 3, 2007, pp. 209 – 218.

Tanenbaum A. (2002). Distributed Systems: Principles and Paradigms. Prentice Hall.

Tsou, M-H. (2003). An Intelligent Software Agent Architecture for Distributed Cartographic Knowledge Bases and Internet Mapping Services. In Maps and the Internet, M. Peterson (Ed.), Elsevier Science, Amsterdam Cambridge, p. 231.

Tyner J. (1992). Introduction to Thematic Cartography. Prentice Hall, Upper Saddle River, New Jersey, 07458.

van Dam, A., Forsberg, A., Laidlaw, D., LaViola, J. and Simpson, R.M. (2000). Immersive VR for scientific visualization: a progress report. Computer Graphics and Applications, IEEE, 20 (6), pp. 26 – 52, http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=888006, accessed 2005-11-22.

Vivid Solutions (2010). JTS Technical Specification. http://www.vividsolutions.com/jts/jtshome.htm, accessed 2010-12-10.

W3C (2003). Scalable Vector Graphics (1.1) Specifications. World Wide Web Consortium.

W3C (2010). Extensible Markup Language (XML). World Wide Web Consortium, http://www.w3.org/XML/, accessed 2010-12-10.

W3C Web Services Architecture Group (2004). W3C Working Group Note 11 February 2004, http://www.w3.org/TR/ws-arch/, accessed 2006-11-20.

Williams, J. and Neumann, A. (2006). Interactive Hiking Map of Yosemite National Park. In Proceedings of the 5th ICA Mountain Cartography Workshop, Bohinj.

WSSN (2007). WSSN World Standards Services Network, http://www.wssn.net/WSSN/aboutwssn.html, accessed 2010-09-13.

Wytzisk, A. (2003). Interoperable Geoinformations- und Simulationsdienste auf Basis internationaler Standards. PhD Thesis, Westfaelische Wilhelms-Universitaet Muenster, p. 134.

Zaslavsky, I. (2003). Online Cartography with XML. In Maps and the Internet, M. Peterson (Ed.), Elsevier Science, Amsterdam Cambridge, p. 171.

Zeder, S. (2009a). Arbeitsablauf für die Erstellung webbasierter Gefahrenkarten auf gemeindeebene. Project Work in GIS and Cartography at MSc. Level. ETH Zurich

Zeder, S. (2009b). QGIS mapserver Manual. ETH Zurich, http://karlinapp.ethz.ch/ qgismapservmanual.pdf, accessed 2010-09-13.

Zhong-Ren Peng, Z.-R. and Chuanrong, Z. (2004). The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS). Journal of Geographical Systems, Volume 6, Number 2, pp. 95-116.

Selected Author's Publications Related to this Thesis

Iosifescu, I. (2007c). OGC CR 07-105: SE Implementation Specification Change Request – extensions for thematic mapping. Open Geospatial Consortium.

Iosifescu, I. (2009). Design of a General Architecture for GIS. MSc. Thesis ETH Zurich, Global Information Systems Group.

Iosifescu, I., Hubentobler, M. and Hurni, L. (2010a). Web cartography with open standards – A solution to cartographic challenges of environmental management. Environmental Modelling & Software 25 (9), pp. 988–999.

Iosifescu, I., Hubentobler, M. and Hurni, L. (2010b). Development of Web GIS and Web Mapping Applications with QGIS mapserver (tutorial reader). In Proceedings of the GIScience 2010 Conference, Zurich, Switzerland.

Iosifescu I. and Hurni L. (2010c). GIS Platform for Interdisciplinary Environmental Research, in Proceedings of the 7th ICA Mountain Cartography Workshop, Borsa, Romania (in press).

Iosifescu, I. and Hurni, L. (2010d). Web-service Driven Cartography, In: Proceedings of the Autocarto 2010 Conference, Orlando, Florida.

Student Works Co-Supervised by the Author

Weber, A. (2006). Interaktive Karte von Methana für mobile Endgeräte. Diploma Thesis ETH Zurich, Institute of Cartography.

Eberle, A. (2009). Visualisierung von Naturgefahrendaten mit QGIS Mapserver und OpenLayers. BSc. Thesis ETH Zurich, Institute of Cartography.

Zeder, S. (2009). Arbeitsablauf für die Erstellung webbasierter Gefahrenkarten auf gemeindeebene. Project Work in GIS and Cartography at MSc. Level, ETH Zurich, Institute of Cartography.

Romer, S. (2010). Kartografische Funktionen in ArcGIS. MSc. Thesis ETH Zurich, Institute of Cartography.

Ortner, F. (2011). Dienstbasierte Kartengenerierung für Web-Atlanten unter Anwendung erweiterter OGC-Standards. MSc. Thesis University of Zurich, Department of Geography, co-supervised by ETH Zurich, Institute of Cartography and Geoinformation.

Curriculum Vitae

Ionut Iosifescu-Enescu
ETH Zurich
Institute of Cartography and Geoinformation
Wolfgang-Pauli-Str. 15
8093 Zurich

Ionut Iosifescu-Enescu is research assistant and PhD candidate at the Institute of Cartography and Geoinformation, ETH Zurich. He graduated in Geodesy (2003) and Spatial Information Systems (2004) at the Technical University of Civil Engineering Bucharest, and obtained a masters degree in Computer Science (2009) from ETH Zurich. In 2004–2005, he collaborated with the project HazNETH at ETH Zurich where he developed Web cartographic applications and geospatial database models for natural hazards. Between 2005 and 2009, he performed research in the frame of the EU FP6 Integrated Projects ORCHESTRA and SANY, where he designed and implemented cartographic web services. Since 2008, he is developing the GIS platform of the COGEAR and SwissExperiment projects and starting 2010 he is contributing to the further development of the GeoVITe geodata distribution portal for the ETH domain. The focus of his research lies in Web Cartography, cartographic web services and geospatial standards.

Personal Data

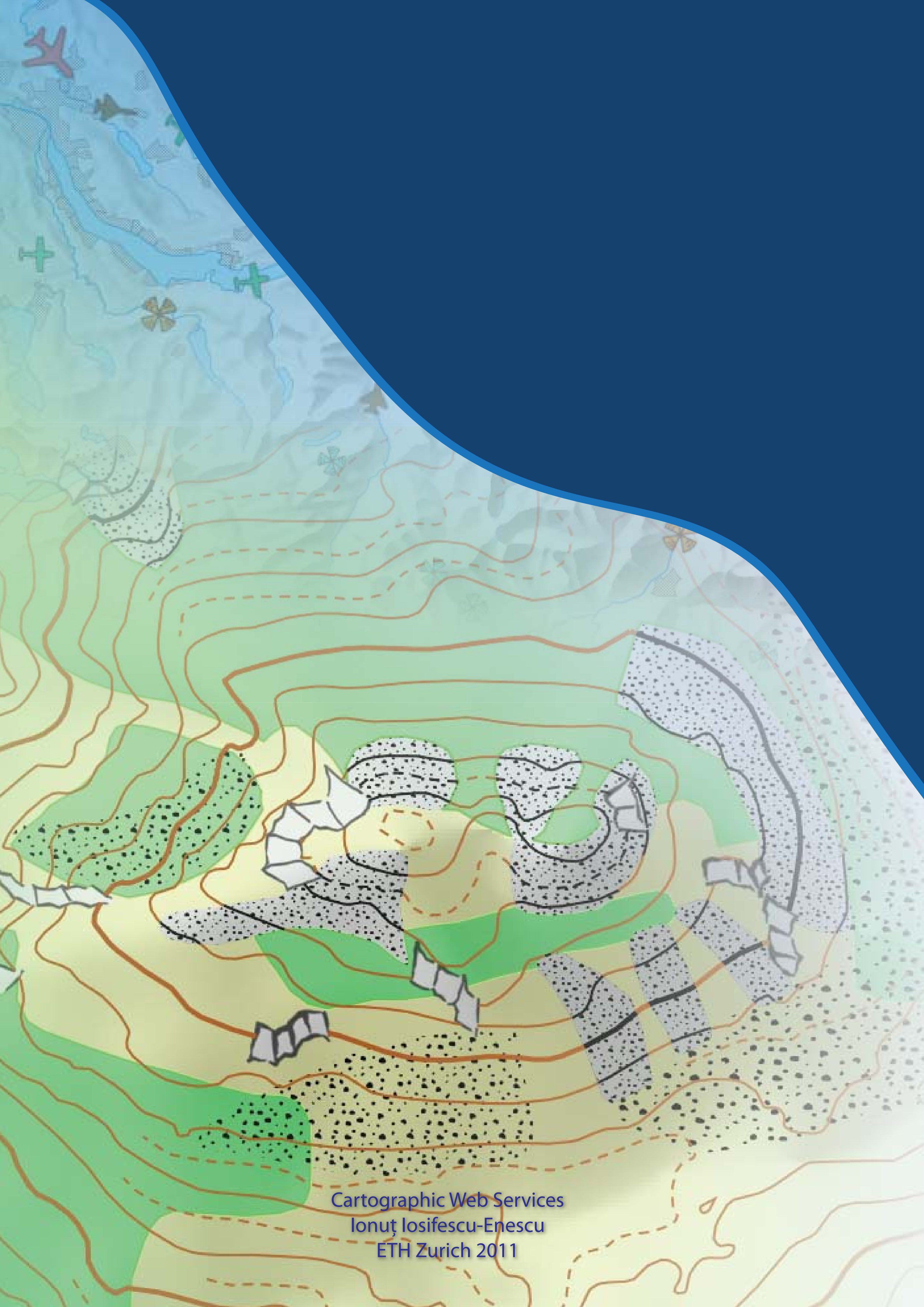
Title	Graduate Engineer in Geodesy, MSc. ETH in Computer Science
Date of Birth	December 20, 1979
Nationality	Romanian

Education

2005–2011	PhD studies at the Institute of Cartography and Geoinformation, Department of Civil, Environmental and Geomatic Engineering, ETH Zurich
2006–2009	Master in Computer Science ETH, Department of Computer Science, ETH Zurich
2005–2006	Certificate of Advanced Studies ETH in Computer Science, Department of Computer Science, ETH Zurich
2004–2005	Certificate of Advanced Studies ETH in Spatial Information Systems, Institute of Geodesy and Photogrammetry, ETH Zurich
2003–2004	Advanced Studies in Spatial Information Systems, Faculty of Geodesy, Technical University of Civil Engineering Bucharest, Romania
1998–2003	Graduate Studies at the Faculty of Geodesy, Technical University of Civil Engineering Bucharest, Romania
1994–1998	High School Certificate, Subject Mathematics-Physics, “Aurel Vlaicu” High School, Breaza, Romania

Work experience

2005–2011	Research assistant at the Institute of Cartography and Geoinformation, ETH Zurich
2004–2005	Swiss federal scholarship at the Institute of Cartography, ETH Zurich



Cartographic Web Services
Ionuț Iosifescu-Enescu
ETH Zurich 2011