Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

RIMapperWMS: a Web Map Service providing SVG maps with a built-in client

Barend Köbben

International Institute for Geo-Information Science and Earth Observation (ITC),
Department of GIP, PO Box 6, 7500 AA Enschede, The Netherlands.
`kobben@itc.nl` – `http://kartoweb.itc.nl/RIMapper/`

Summary. This paper introduces RIMapperWMS, a light-weight web mapping application that conforms to the Open Geospatial Consortium's Web Map Service specification. It serves interactive web maps from a spatial database back-end. Compared to existing WMS implementations, it stands out firstly because it serves its maps in the Scalable Vector Graphics format. This allows it to offer high-quality vector cartography, specially suitable for mobile devices such as PDA's and smartphones. Secondly, RIMapperWMS includes so-called VendorSpecific Capabilities that the OGC specification allows, in this case enabling it to produce the SVG output with a built-in Graphical User Interface, allowing the data to be disseminated to any SVG-capable application, without the need for a separate WMS client.

Key words: Web Mapping, Spatial Databases, WMS, SVG, built–in GUI

1 Introduction

This paper introduces RIMapperWMS, the result of a project to build a light-weight web mapping application that conforms to the Open Geospatial Consortium's (OGC) Web Map Service (WMS) specification. We started out this project with two main questions: Firstly to see if it would be possible and practical to extend earlier work in our SDI^{light}project to build a WMS that serves its maps in the Scalable Vector Graphics format (SVG). This would allow it to offer high-quality vector cartography, specially suitable for mobile devices such as PDA's and smartphones. Secondly, we wanted to investigate the possibility for this WMS to produce the SVG output with a built-in Graphical User Interface (GUI). In this way, the data could be disseminated to any SVG-capable application, without the need for a separate WMS client, something no existing WMS offers at present.

The resulting RIMapperWMS system serves interactive web maps from a spatial database back-end. The general components of the RIMapperWMS setup, as shown in figure 1, are:

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

- A spatial database back-end is used for storing both the configuration of the Web Map Services as well as the actual spatial and attribute data the maps are derived from. The former is stored in normal attribute tables, the latter using the OGC Simple Features spatial tables.
- A set of Java servlets and classes that respond to WMS compliant request from wireless or wired web clients by providing maps in SVG, with a built-in GUI that consequently can be used to generate further request to the WMS, for zooming, panning, information retrieval, etcetera.
- A mobile or desktop web client capable of rendering SVG to view and interact with the maps.

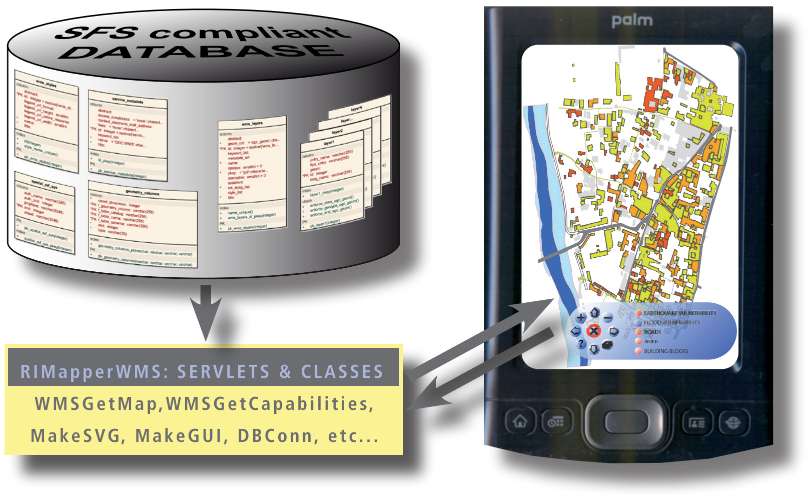
SFS compliant DATABASE
palm
RIMapperWMS: SERVLETS & CLASSES
WMSGetMap, WMSGetCapabilities,
MakeSVG, MakeGUI, DBConn, etc...

Fig. 1. General Principle: RIMapperWMS server components use the configuration settings and Simple Features for SQL (SFS) compliant data in a spatial database back-end to deliver an SVG map, including a Graphic User Interface (GUI) to a (mobile) client.

As mentioned above, RIMapperWMS is based on the SDI^{light}concept and has a range of precursor projects, briefly introduced in section 2. The rationale of developing further into RIMapperWMS is discussed in section 3.1. The technical setup is described in section 3.2, together with a discussion on the consequences of adding a GUI to the map output for specification compliance as well as for interoperability. Finally the current status of the project will be presented in section 3.3 and some conclusions and an outlook to further development will be offered in section 4.

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

2 Earlier Projects at the Basis of RIMapperWMS

RIMapperWMS has grown out of a number of earlier projects. These employed many of the software and database solutions used later for RIMapperWMS, and they have greatly influenced the way it has been set up, therefore we will briefly describe these projects first. Like many projects and educational materials developed within the GeoInformation Processing Department of the International Institute for Geo-Information Science and Earth Observation (ITC), they were based on what we dubbed the 'SDI^{light}' philosophy.

2.1 SDI^{light}

The term Spatial Data Infrastructure (SDI, sometimes also GDI for GeoData Infrastructure) may be usually connected with (very) large regional or national spatial data warehouses, but it is defined more generally in [3] as "the networked geospatial databases and data handling facilities, the complex of institutional, organizational, technological, human and economic resources (...) facilitating the sharing, access to, and responsible use of geospatial data at an affordable cost for a specific application domain or enterprise." In many cases SDI data and application infrastructures are being developed using high-end geospatial software solutions and large corporate databases, needing substantial investments in financial and human resources. But the principles of SDIs can be applied in more simple and cost-effective ways just as well. This down-to-earth approach, which we named SDI^{light}, is of particular interest for students, partners and clients of the International Institute for Geo-Information Science and Earth Observation (ITC), an institute that aims at capacity building and institutional development specifically in developing countries.

SDI^{light}serves as a general purpose test bed for applied as well as fundamental research activities, and should provide researchers and students alike with a proof-of-concept platform for relatively simple, low-cost, yet powerful ways of sharing data amongst various distributed offices and institutions as well as the general public. To achieve that, we use Open Standards whenever available, Open Source solutions where possible and commercial software where necessary. And because a considerable part of our courses is aimed at training geo-informatics engineers, we have students also actively develop and build (parts of) such systems. In general, the main building blocks are:

- *A spatial database backend* that stores the geometry and the attribute data; Spatial data is stored using the Open Geospatial Consortium (OGC) Simple Features specifications. Both PostgreSQL/PostGIS and MySQL have been used.
- A set of *interoperable web applications* that interface with the database and with each other, and fulfil tasks such as delivering maps for visualisation purposes, provide data in formats such as GML for data exchange, etcetera. We either use the UMN MapServer software, or develop our own components in Java, using Apache Tomcat for deployment.

- Simple *Web-based interfaces* enabling access to the maps and data for both desktop browsers and mobile platforms. At present, we concentrate on web browser clients, employing various techniques such as Dynamic HTML (DHTML), Asynchronous JavaScript And XML (AJAX) and Scalable Vector Graphics (SVG).

It is not the intention that SDI^{light}should finally become one coherent system, rather it should be seen as a test bed in the broad sense of *equipment for testing*. It is the place where we can show fellow researchers, consultants and students as well as possible users (such as GIS and Cartography departments in developing countries) that the things we teach can be made to work quite quickly, in a relatively simple and low-cost setup. As such it has been functioning for some three years now and has spawned a variety of projects, of which RIMapperWMS is the latest outcome.

2.2 RIMapper

A first application called RIMapper was developed in 2003 to investigate the possibility of generating light-weight, versatile Risk Indicator Maps (RIMs) and deliver these to web clients as interactive SVG maps, based on XML configuration files. These maps were to be part of an urban risk management system, and therefore needed to fit a multitude of use cases, ranging from giving the general public information about risks, to providing local authorities an interface to the underlying risk assessment databases and models. Furthermore, the maps needed to be usable on a wide range of platforms, from the office systems of the local authorities to hand-held devices providing location based services to field personnel.

In the database tier, geometric features are stored as OGC Simple Features geometry using a MySQL database. Map layers can be either styled uniformly, eg. all roads sharing the same visualisation, or depending on some data attribute per feature, eg. for a chorochromatic map of homes viewed by vulnerability type. In RIMapper, the database also stores common SVG code fragments and ECMAscript event listeners.

The application tier is a set of generic Java classes to do recurring tasks like extracting OGC features and attribute data from the database, translating these into fragments of SVG and ECMAscript, collecting and structuring these fragments into valid output and delivering this output to the clients. The glue provided to make all these parts act together are the XML map configurations. They are parsed to get a description of the map needed and all its component parts, as well as the choice for the visualisation type per layer.

When all data needed has been collected by the system, the SVG output is composed and handed over to the web server for delivery to the client. The SVG generated will adhere to the SVG 1.1 Basic profile and will thus be suited for a broad range of clients, including PDA's.

Using the RIMapper system, described in more detail in [6], one can very flexibly offer database-driven maps on the web that are generated on the fly

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

from the most recent data, and that can incorporate all the functionality, scalability and graphics quality that the SVG standard offers. The main disadvantage of this original system is that it outputs the whole data extent in one client-side SVG file. This simplifies setup and is no problem for the small risk indicator maps it was designed for, but it makes the system not well scaleable. Furthermore, the service interface does not comply to any of the available Open Standards.

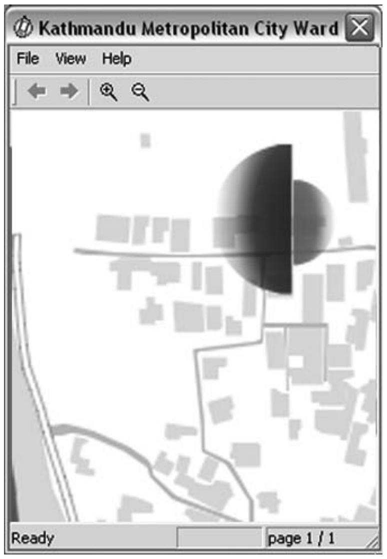

Fig. 2. Example of a RIMapper interactive SVG map. It features a dynamically changing risk symbol showing the modelled risks (left half for floods, right half for earthquakes) at the current pointer location (screen dump from BitFlash SVG mobile viewer on PDA, interactive original available on website [5]).

2.3 Web Application for Landslide Inventory

A second application of the RIMapper technology was a pilot of a web-based system for updating the landslide map of the Serchio basin (Central Tuscany, Italy). This is an official document that represent the actual state of the landslides in the region. The updating of the map is at present carried out by the local municipalities using paper sketch maps. The objective was to significantly speed up and simplify this updating process, while taking into account the severe constraints of the municipal organisations. This was to be achieved by providing them with a lightweight Web based map application that allows inventory of new landslides and submitting them via the WWW directly to a central database.

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

The functionality of RIMapper had to be extended with a mechanism for input of the landslide locations. Implementing a digitizing functionality proved quite straightforward, using the scripting possibilities of SVG. The next step is to insert a new path element in the DOM-tree. The client-side functionality for upload of the newly digitized polygon to the database is implemented, but the authorities still will need to rework the server-side to actually receive the inputs and store them in a safe, transactionally sound, manner.

The work described here (and further in [9]) highlighted some of the inherent weak points of the original RIMapper setup. Having the landslide map for the whole Serchio basin client-side, proved to be inefficient indeed. A system where only the data needed are loaded from the server would be required for a production version. Such a system was also needed and indeed devised for the Wireless Campus projects introduced below.

2.4 Wireless Campus LBS and CampusMapper

The Wireless Campus LBS project, described in more detail in [8], is a co-operation between ITC and the University of Twente (UT). Its aim is to provide a platform for Location Based Services (LBSes) for the UT campus. The foundation for these LBSes is the existing Wireless Campus system that provides the whole University grounds with WiFi based internet access. The purpose of the project is not the development of *the* or even *a* Wireless Campus LBS, but rather to investigate and set up the infrastructure necessary for LBSes based on it. It combines input from several research projects, eg. on WiFi positioning techniques [10] and on context-aware databases [1], with the practical application of new as well as established techniques to provide useful services for the UT campus population, and is intended to serve as *a test bed for* research as well as *to benefit from* the outcomes of research.

A first result was an LBS for participants of SVG Open 2005 (the 4th Annual Conference on Scalable Vector Graphics), that was organised at the UT in August 2005. This pilot application called FLAVOUR (Friendly Location-aware conference Assistant with priVacy Observant architectURe, see [11] and figure 3) used WiFi localization techniques to firstly offer *pull services* for general navigation as well as for locating fellow attendants and conference facilities. Secondly, *push services* were established, eg. for messenger-type peer-to-peer communication and notifications by conference organizers. The tests at SVG Open 2005 were relatively successful: The localization functionality worked quite reliably, although the accuracy was varying quite a bit over the various conference locations.

The *mapping service* of FLAVOUR (see figure 3) was based on RIMapper, but this time the server application was set up to request only those parts from the spatial database that are needed for the current map extent. This change was unavoidable, because for this application a fairly large database has been built, incorporating large scale topographic and building data of the whole 140 hectare campus and its 650+ wireless network access points.

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

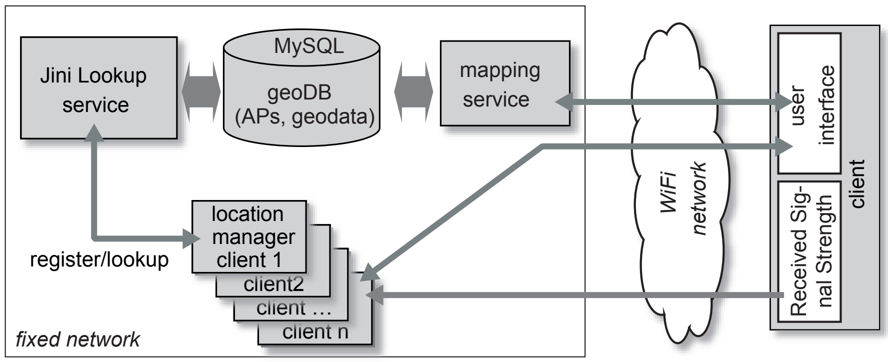

Fig. 3. Setup of the FLAVOUR prototype (after [11]).

The database built for the Wireless Campus LBS can serve more purposes then the WiFi localization only. The University Service departments (such as Public Relations and Facility Management) are especially interested in it, as it can provide the basis for easily customisable maps of the campus or parts thereof. We helped this idea forward by building a small prototype of such an interactive, web-based application, called CampusMapper [7]. The client-side maps are again generated as SVG, based on the bounding box requested from the database by the application tier. CampusMapper uses DHTML for the user interface part of the client (for choosing the layers, visualisation type, zooming, panning, etcetera), instead of the XML configuration files of earlier projects. These configuration settings are communicated to the application tier as parameters of the HTTP-GET QueryString object. The CampusMapper pilot is currently being further developed in several student projects, to hopefully lead to an implemented version on the University website.

3 RIMapperWMS

With the changes to the original RIMapper application tier as described in 2.3 and 2.4, its functionality had by now become very similar to that of an OGC Web Map Service. It therefore seemed logical to take a next step of making its behaviour actually conform to the full WMS specification.

3.1 Rationale and setup principle

The Open Geospatial Consortium's Web Map Service (WMS) specification [4] is no doubt the most widely implemented of OGC's Open Standards. Nowadays most major commercial GIS vendors offer WMS capabilities and there is Open Source and free software available that does the same. In July 2006, the

OGC Implementation Statistics web page [14] listed 146 implementations. Although these generally render their maps in a pictorial format such as PNG, GIF or JPEG, a small proportion of them also offer vector-based maps in Scalable Vector Graphics (SVG). Therefore, on first sight it may seem that a WMS version of RIMapper would not add much to already available solutions.

However, it is our impression that existing implementations that do support SVG all treat the output as just another graphics format, and like the GIF and JPEG output, the maps are basically pictures only, with no interactivity or 'intelligence'. This impression is based on our intimate working knowledge with two of the most prominent WMSes to support SVG (UMN Mapserver and GeoServer) and a quick scan of the advertised capabilities of the other WMS implementations as found through URL [13]. To have the map behave like a *mapping application*, it needs to be wrapped in a Graphical User Interface (GUI). There are many such GUI's available, as dedicated server-side applications, or as part of general GIS clients (such as the Java-based freewares uDig and Jump).

But SVG is more than a graphics format only, it also offers interactivity, through built-in ECMAscripting and full access to its XML Document Object Model. It is therefore possible to build an SVG map, or rather an *SVG application*, that includes its own GUI, which has been demonstrated by various examples available (for some excellent examples and tutorials, see URL [2]). These solutions are programmed on a case-to-case basis however, and not easily generated or deployed repeatedly from a dynamic set of data. RIMapperWMS was set up to make such a solution more generic, by offering a simple WMS conformant interface to the spatial data, including its own built-in client-side GUI. This GUI handles the map interaction and based on that will generate new WMS-conformant service requests to change the view on the data, add or remove data layers, get information on data attributes, etcetera. This principle is outlined in the UML Sequence Diagram in figure 4.

At first sight, it might seem that including the GUI generation part in existing open source software would have been more straightforward than developing our own WMS implementation. There were two main reasons to do so anyway: We did not have to build in from the ground up, as the existing RIMapper system already supported almost all functionality required. Making it behave WMS compliant was estimated a limited effort, and indeed took less than 1 man–month. Additionally, in this way the system remains light-weight: only the functionality needed is in the system (the spatial DB, the WMS server and SVG client logic), without the overhead of more generic systems such as UMN MapServer, which also support many other OGC specifications, datastores and output formats.

3.2 Technical Implementation

The database tier of RIMapperWMS is implemented in the PostgreSQL database server. When using the PostGIS spatial extension for this Open

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

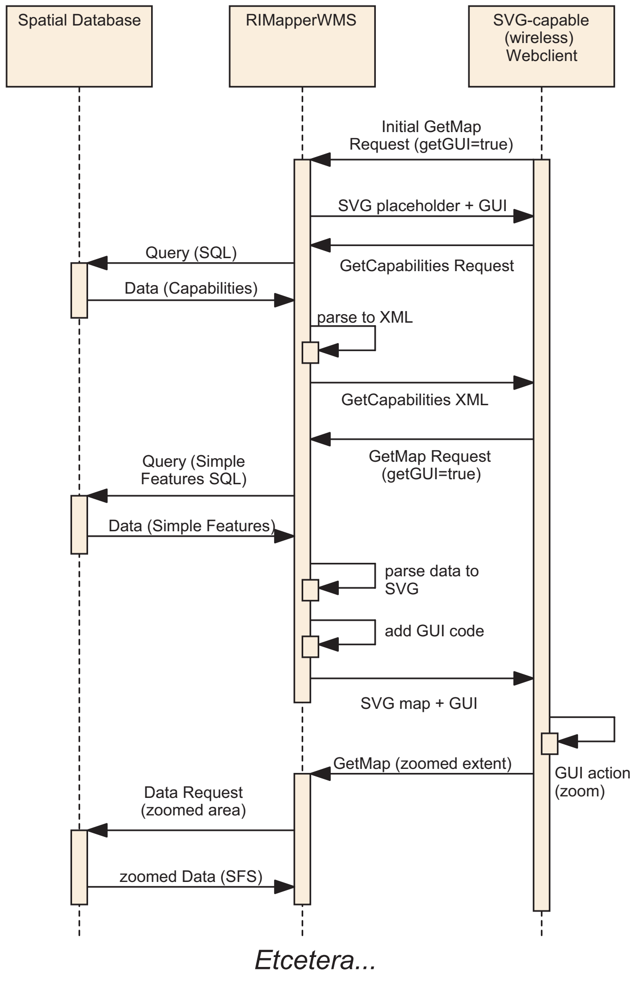

Fig. 4. UML Sequence Diagram of requesting an SVG map with a built-in GUI (using the `getGUI=true` parameter) and then zooming in.

Source DBMS, one has a powerful, fully OGC-compliant spatial data server. In earlier projects, we also supported the use of the MySQL database. MySQL's recent versions do include spatial extensions, but they are not implementing the full OGC specification, and especially the lack of support for Co-ordinate Reference Systems and therefore for (re)projection and transformation, makes it unsuitable for a WMS. In the future, connectors for other compliant datastores could be added.

All configuration is done using database tables. The system can simultaneously serve multiple WMS 'instances', each from a seperate database. As

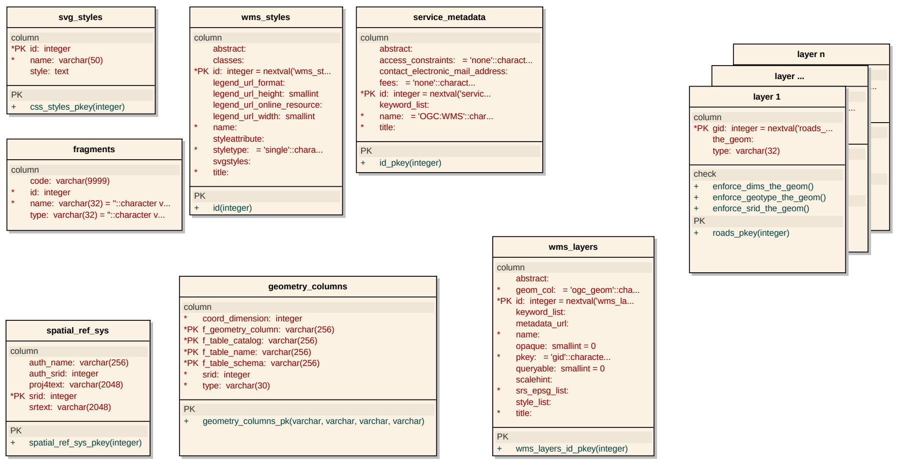
svg_styles
column
*PK id: integer
* name: varchar(50)
style: text
PK
+ css_styles_pkey(integer)
wms_styles
column
abstract:
classes:
*PK id: integer = nextval('wms_st...
legend_url_format:
legend_url_height: smallint
legend_url_online_resource:
legend_url_width: smallint
* name:
styleattribute:
* styletype: = 'single'::chara...
svgstyles:
* title:
PK
+ id(integer)
service_metadata
column
abstract:
access_constraints: = 'none'::charact...
contact_electronic_mail_address:
fees: = 'none'::charact...
*PK id: integer = nextval('servic...
keyword_list:
* name: = 'OGC:WMS'::char...
* title:
PK
+ id_pkey(integer)
fragments
column
code: varchar(9999)
* id: integer
* name: varchar(32) = ''::character v...
type: varchar(32) = ''::character v...
layer n
layer ...
layer 1
column
*PK gid: integer = nextval('roads_...
the_geom:
type: varchar(32)
check
+ enforce_dims_the_geom()
+ enforce_geotype_the_geom()
+ enforce_srid_the_geom()
PK
+ roads_pkey(integer)
wms_layers
column
abstract:
* geom_col: = 'ogc_geom'::cha...
*PK id: integer = nextval('wms_la...
keyword_list:
metadata_url:
* name:
opaque: smallint = 0
* pkey: = 'gid'::characte...
queryable: smallint = 0
scalehint:
* srs_epsg_list:
style_list:
* title:
PK
+ wms_layers_id_pkey(integer)
geometry_columns
column
* coord_dimension: integer
*PK f_geometry_column: varchar(256)
*PK f_table_catalog: varchar(256)
*PK f_table_name: varchar(256)
*PK f_table_schema: varchar(256)
* srid: integer
* type: varchar(30)
PK
+ geometry_columns_pk(varchar, varchar, varchar, varchar)
spatial_ref_sys
column
auth_name: varchar(256)
auth_srid: integer
proj4text: varchar(2048)
*PK srid: integer
srtext: varchar(2048)
PK
+ spatial_ref_sys_pkey(integer)

Fig. 5. UML object diagram of the Data Model for the spatial data back-end. Note the *central* wms_layers table, *n* layer tables on the *right*, further WMS configuration tables on the *upper-left* and PostGIS-specific tables in the *lower-left* side.

can be seen in the data model in figure 5, the central configuration table is `wms_layers`. This lists which data sets are available, what their co-ordinate system is, what styles they support, etcetera. The actual layer data is stored in n layer tables, that include the OGC compatible geometry columns and any attributes available. These layer tables are usually the result of imports from other data sources, eg. from ESRI shape files through the PostGIS loader tool. Further WMS configuration is taken care of in the `service_metadata`, `wms_styles` and `svg_styles` tables. All this information is used by the application tier to determine the capabilities of the WMS instance. Setting these configuration tables is currently done by using a generic SQL client, but we plan for future web-based configuration pages to make this process more user-friendly.

The application tier is employing a set of Java servlets and classes that can be deployed in any J2EE compatible servlet container (eg. Apache Tomcat). The functionality of the previous projects has been expanded to support the interfaces required for a *Basic WMS*: `GetCapabilities` and `GetMap`. `GetCapabilities` returns an XML description of the WMS's information content and acceptable request parameters. `GetMap` returns the map itself. The WMS client can specify which information to be shown on the map (one or more Layers), the Styles of those Layers, what portion of the earth is to be mapped (Bounding Box), the projected or geographic co-ordinate reference system to be used (the Spatial Reference System), the desired output format, the output size, etcetera. The WMS specification also describes an optional *Queryable WMS*, additionally supporting the `GetFeatureInfo` interface, used for retrieving information about particular features shown on a map.

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

Although our built-in GUI supports getting feature information client-side, by mouse-clicks or mouse-overs, it is not implemented through the standardized `GetFeatureInfo` interface at the moment.

The client-side software needed for this setup has to be able to issue HTTP-GET and/or POST requests, as well as being able to render SVG content. With SVG being an official W3C recommendation, theoretically any web browser would be sufficient, but in practice the situation is somewhat more complex. The most-used browser, Microsoft Internet Explorer, currently needs a plug-in for supporting SVG. Some browsers have implemented native (inline) SVG support, with Firefox (1.5+) and Opera (9+) well on their way to support the full standard, and our maps work well in both. The fact that Vista, the new Microsoft operating system, uses a graphics layer (XAML) that is based on SVG, has lead to speculation that upcoming IE versions will also support inline SVG. Furthermore, there are several stand-alone SVG clients, especially for PDA's and smartphones.

3.3 Status of the current implementation

The first public bèta of the system was released in December 2006 on the website `http://kartoweb.itc.nl/RIMapper/`. This first release supports the functionality required for a *Basic* WMS version 1.1.1. Currently a newer WMS 1.3.0 specification is ready, but there has not been a wide-spread adoption of that specification. The main reason seems to be that it included some major changes in the underlying concepts, specifically in the way co-ordinate reference systems are dealt with. It requires, among other things, that the system can handle data an arbitrary sequence of axes for a projection, as opposed to the fixed longitude–latitude order of the 1.1.1 specification. Our system uses the transformation and projection capabilities of the underlying PostGIS database, which in turn uses the open source PROJ4 library [15], as do UMN Mapserver and many other projects. This means that until that library has been upgraded to allow the new projection system, our and many other WMS servers will not be able to support the new specification.

A point of discussion in implementing the system has been the question of possibly breaking the interoperability principle of the WMS with our additions. In our system, requesting the built-in GUI is done by adding a specific parameter (`getGUI=true`, see figure 4). Such an addition is covered by the specification in the form of so-called VendorSpecific Capabilities. However, our specific addition introduces a WMS layer that has a built-in client which is intrinsically unaware of any additional layers in a possible stack of so-called *cascaded* WMS layers. In other words, when this RIMapperWMS layer is requested by a different WMS client, or as input for another WMS service, any GUI actions from our layer would break the interoperability chain. For that reason, the GUI is only included if the request comes from a RIMapperWMS client, and therefore includes the VendorSpecific `getGUI=true` parameter. If not, the GUI is not included and the map behaves like any other basic WMS

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

layer. In that case the output format could also be something other than SVG, because we could use a transcoder to convert the SVG into PDF, JPEG, GIF, TIFF or PNG. This alternative setup is depicted in figure 6.

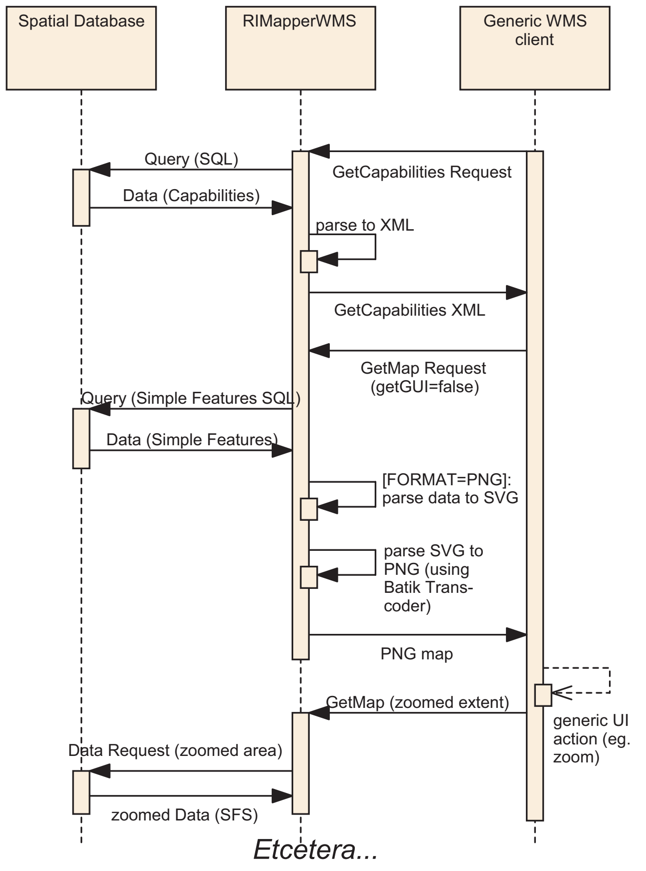

Fig. 6. UML Sequence Diagram of requesting a standard WMS map without GUI (`getGUI=false`), and using the PNG format instead of SVG.

At present, the bèta is fully functional as a Basic WMS 1.1.1 server and has been tested using our own data as well as the test data set from the OGC Compliance Testing pages [12]. Most OGC tests were passed, but full compliance cannot be achieved yet, mainly because that requires PNG or GIF output and we have yet to implement the transcoding mentioned above. We are not an exception here, as out of the 146 WMS implementations mentioned earlier, only some 22 claim full 1.1.1 compliance (UMN Mapserver, one of the most used WMSes, not being one of them!).

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

As all components of the system are either developed by ourselves, or existing Open Source software, we are able to offer the whole system in a simple-to-deploy package, with installation instructions and example data. The system is made available under an Open Source license and we invite anyone interested to experiment with the system, provide us with feedback or further develop the functionality.

4 Conclusion and Outlook

This project made it clear that RIMapperWMS was a logical extension of the series of software projects that all are based on the SDI$^{\text{light}}$philosophy explained in the section 2. These projects have served well as a test-bed and proof-of-concept platform for a limited group of students and researchers. Because this latest branch on the tree is implementing the popular Open Standard WMS, we hope it might also be used in a broader context, by persons and institutions that have a need for a relatively light-weight, yet reasonably powerful way of disseminating their spatial data through the Web.

Furthermore, we conclude that making the WMS produce SVG output with a built-in Graphical User Interface is possible within the OGC specification and indeed makes the output light-weight and easily deployed.

We plan to further develop RIMapperWMS. Extending the functionality to a *Queryable WMS*, by adding support of the `GetFeatureInfo` interface, is high on the wish-list, other plans include support for client-side or server-side Styled Layer Descriptors and Web Map Context Documents. Depending on the developments of the underlying libraries, we also hope to be able to include WMS 1.3.0 support at a later date.

When the application is considered stable, we plan to do useability and performance tests of our architecture, especially for use on mobile devices. Included in these tests should be a comparison between our strategy of including the GUI in the SVG response with the more 'typical' approach of having a stand-alone SVG capable client consuming the WMS.

Acknowledgments. As mentioned earlier, RIMapperWMS grew out of a number of projects, and the students and researchers involved are gratefully acknowledged for their contributions: Georgi Koprinkov, Stephanie Krane, Maurizio Latini, Nirvana Meratnia, Kavitha Muthukrisnan, Sebastiaan Waanders.

References

1. Arthur H. van Bunningen, Ling Feng, and Peter M.G. Apers. Context for ubiquitous data management. In Katsumi Tanaka, Yutaka Kidawara, and Koji Zettsu, editors, *the 2005 International Workshop on Ubiquitous Data Management (UDM'05)*, pages 17–24, Tokyo, 2005. The IEEE Computer Society.

Published in:
The European Information Society - Leading the way with Geo-information
(Lecture Notes in Geoinformation and Cartography),
S. Fabrikant & M. Wachowicz (eds.). Springer-Verlag: Berlin, etc. pp. 217-230.

2. Carto:Net. SVG navigation tools tutorial, Last accessed: March 2006. `http://www.carto.net/papers/svg/samples/`.
3. Richard Groot and J. McLaughlin. *Geospatial data infrastructure; concepts, cases and good practice*. Oxford University Press, Oxford, etc., 2000.
4. Open Geospatial Consortium Inc. Web Map Service Implementation Specification (1.1.1). Technical Report 01–068r3, OGC, 2002.
5. International Institute for Geo-Information Science and Earth Observation (ITC). RIMapper project pages, Last accessed: February 2007. `http://kartoweb.itc.nl/rimapper/`.
6. Barend Köbben. RIMapper—a test bed for online Risk Indicator Maps using data-driven SVG visualisation. In Georg Gartner, editor, *2nd Symposium on Location Based Services and TeleCartography*, Geowissenschaftliche Mitteilungen, pages 189–195, Wien, 2004. Institute of Cartography and Geo-Media Techniques.
7. Barend Köbben and Stephanie Krane. CampusMapper—a light-weight internet mapping tool using MySQL, Tomcat and SVG, Last accessed: September 2006. Unpaginated CD and website: `http://www.foss4g2006.org/`.
8. Barend Köbben, Kavitha Muthukrisnan, Nirvana Meratnia, and Georgi Koprinkov. Wireless Campus LBS—A testbed for cartographically aware database objects. In Georg Gartner, editor, *Symposium 2005 Location Based Services & TeleCartography*, Geowissenschaftliche Mitteilungen, pages 47–51, Wien, 2005. Research group Cartography, Institute of Geoinformation and Cartography, TU Wien.
9. Maurizio Latini and Barend Köbben. A web application for landslide inventory using data-driven SVG. In Peter van Oosterom, Siyka Zlatanova, and Elfriede M. Fendel, editors, *the 1st international symposium on Geo-information for Disaster Management*, pages 1041–1054, Berlin, etc., 2005. Springer-Verlag.
10. Kavitha Muthukrishnan, Maria Eva Lijding, and Paul Havinga. Towards Smart Surroundings: Enabling Techniques and Technologies for Localization. In *LOCA2005 – co-allocated with the 3rd International Conference on Pervasive Computing*, page 11, Munich, 2005. Springer Verlag.
11. Kavitha Muthukrishnan, Nirvana Meratnia, and Maria Lijding. FLAVOUR—Friendly Location-aware conference Aid with priVacy Observant architectURe. Technical Report TR–CTIT–05–28, University of Twente, CTIT, 2005.
12. Open Geospatial Consortium Inc. Compliance Testing page, Last accesed: February 2007. `http://www.opengeospatial.org/resource/testing/`.
13. Open Geospatial Consortium Inc. Implementation by Specification page, Last accesed: February 2007. `http://www.opengeospatial.org/resource-/products/byspec`.
14. Open Geospatial Consortium Inc. Implementation Statistics page, Last accesed: July 2006. `http://www.opengeospatial.org/resource/products/stats`.
15. Frank Warmerdam. PROJ4. Cartographics Projections Library, Last accessed: December 2006. `http://www.remotesensing.org/proj/`.