Efficient Integration of Spatial Data into Modern Web-Applications

Bernd Müller, Björn Voigt, and Harald Wehr

University of Applied Studies and Research
Friedrichstr. 57-59,
38855 Wernigerode, Germany
{bmueller,bvoigt,hwehr}@hs-harz.de

Abstract. Nowadays almost all companies, public authorities or other institutions have developed some kind of web-application to present their information to dedicated end users. The different implementations range from simple static HTML pages to dynamically information generated for each request at runtime. A lot of applications additionally need to integrate spatial data. As these kind of information are somehow different from other data, the integration, administration and presentation is not as easy as in traditional web applications. This paper gives an overview how we handle spatial information in the MobiHarz project [1]. As this architecture completely consists of free software, parts of the system can easily be adopted to other spatial applications on the web. Especially public authorities shall be encouraged by this article to use freely available software in spatial information systems.

1 Introduction

By the end of 2002 about 600 million people are using the Internet regularly. A huge amount of static and dynamic web sites provided the users with the requested information. Data stored in databases or other data sources can easily be integrated into the generation of HTML- or WML-pages via certain web applications. Besides this 'traditional' data there is an increasing demand for efficient administration and integration of spatial information into web applications. Due to the special nature of spatial data this is not a trivial task. In this paper we propose a scalable and flexible solution to the depicted problem. The different software components we are using are all freely available and can be configured or modified to certain needs. Besides several programs from the GIS community[2] other free software like *Cocoon* or *PostgreSQL* provide the framework for a web application, that is able to handle spatial information.

The remainder of this paper is organized as follows: in the next section we summarize the characteristics of spatial data that lead to a special handling

[1] The project MobiHarz is part of the global research program "Freizeitverkehr" sponsored by the German Federal Ministry of Education and Research.

[2] For an overview of the most important free GIS software see [4].

of this kind of information. Afterwards we describe our architecture in general. Some of the more important components are explained in more detail. Finally we conclude our current work and give an outline of what will be done in the near future.

2 Special Features of Spatial Information

Describing things of our world is done by abstraction and generalization of existing objects into simpler representatives that can be queried and stored in common databases. Spatial information appears in two different forms:

- continuous phenomena that appears unlimited in areas and space (e.g. temperatures at certain heights),
- discrete objects and fields like areas or line-based objects (e.g. seas or streets).

These spatial information are almost all pictured by either raster or vector data [8]. Each geographic information consists of certain place data, an optional time attribute and further optional attributes that give detailed information about the described point on the earth.

2.1 Raster Data

Raster information are used for describing continuous phenomena. A raster file consists of a matrix of square cells. Every cell represents some point on the earth and gets the corresponding attributes assigned [10]. Depending on the scale of the raster map each pixel can cover a meter up to several kilometers of the earth's surface. Figure 1 shows how raster information is typically represented in

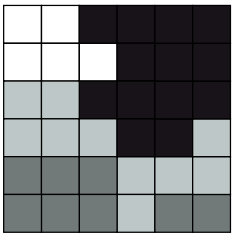


Fig. 1. Representation of Raster Information

a Geographic Information System (GIS). It is mainly used for map and satellite images or scanned pictures of maps. Unfortunately these files can end up in very large data sets. Another disadvantage of raster information are the possible aliasing effects at certain resolutions. Nevertheless the matrix format allows the

user to combine several raster maps to new data sets through geometric and logic operations. The resulting cells only depend on the relevant source cells at the specified geographic locations and the calculation within the used formula [1][8].

2.2 Vector Data

Vector data store information about line phenomena or areas that consists of several closed lines. The vector model provides three geometric elements to describe discrete characteristics of the earth:

- The **point** is used to depict a specific location on the earth's surface. Additionally it links two lines.
- **Lines** are used to represent objects of the real world that can be generalized to a line. These are for instance streets or service pipes. They are also used as borderlines for closed polygons.
- Several closed lines are combined to **areas** to depict closed objects of the earth like oceans, seas or woodland.

Vector data allows exact form descriptions of real world objects. Information is not restricted to a certain scale of the map. Vector information result in very small data sets as only line and vertex information and their corresponding attributes have to be stored. Nevertheless this data format is not as appropriate

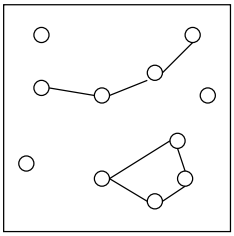


Fig. 2. Representation of Vector Information

as raster information for generating images or elevation data. Vector data is mainly used for describing discrete things of our world.

2.3 Geographic Data Formats

As shown in the previous two sections raster and vector data are used to describe things of the earth. Unfortunately there are lot of different formats in which these information can be encoded [2]. In the past years several GIS-companies, public

Table 1. Some Popular Data Formats for Spatial Information [8]

Vector Data	**Raster Data**
Encapsulated Postscript (EPS)	Tag Image File Format (TIFF, GeoTIFF)
ESRI Shapefile (SHP)	Portable Network Graphics (PNG)
AutoCAD Drawing Exchange Format (DXF)	PC Paintbruch Exchange (PCX)
Spatial Data Transfer Standard (SDTS)	Digital Elevation Model (DEM)

authorities and countries often developed their own format of storing spatial information. Table 1 shows the most popular data formats today.

To support interoperable solutions between the different systems and data formats the Open GIS Consortium (OGC) tries to develop adequate interfaces and protocols. OGC is an international industrial consortium of about 250 companies, government agencies and universities. [11] gives an overview of current specifications, initiatives and requests for comments or papers. It is important to know, that there are already some implementations of the OGC-specifications available. Within the MobiHarz system all vector information are stored in a PostgreSQL database following the *OpenGIS® Simple Features Specification for SQL*. This specification defines a standard SQL schema supporting storage, retrieval, query and update of spatial features. It is implemented by the *PostGIS* project [13] described in the next section. This is the basis for sharing common data between different applications. Furthermore it is easier to reuse the collected spatial information in the future as there are no dependencies on proprietary data formats from specific software vendors.

3 Architecture

The overall architecture of the MobiHarz system is depicted in figure 3. It consists of many different GIS and non-GIS software components. Again, all software is freely available and heavily supported by an enthusiastic community all over the world. Some important parts of the system are now explained in more detail.

3.1 Cocoon

The heart of the system is *Cocoon*, a publishing framework written in Java [3]. Cocoon can be used as a normal Java program via the command line interface but is typically deployed as a servlet in a servlet container. It is therefore responsible for the whole communication with all users. As *Cocoon* is based on strict separation of content, logic and style integration of external components is very easy. This is very important as the framework does not contain any GIS functionality from scratch.

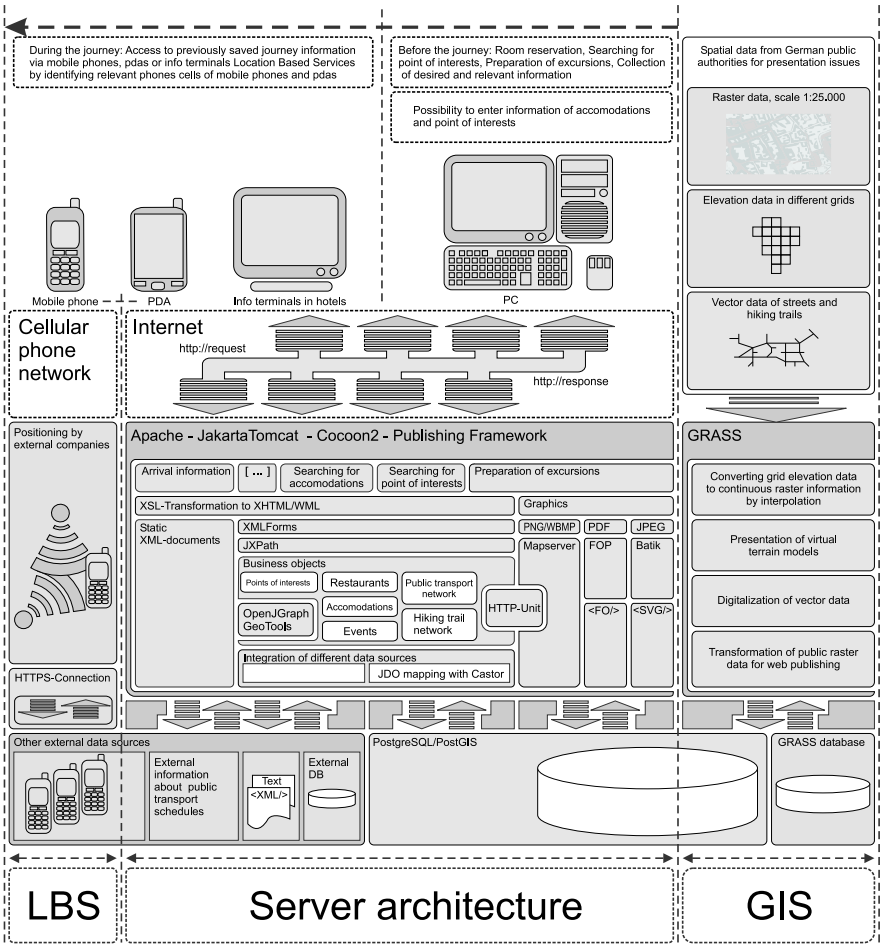


Fig. 3. MobiHarz Architecture of a Spatial Web Application

The main concept of *Cocoon* is based on pipelines. According to an incoming request the appropriate pipeline of the system is called. Each pipe consists of a required generator, optional transformers and finally a required serializer. A generator produces the desired output as normal XML-document. It aggregates content from the filesystem, databases, HTTP-requests or other sources. Following the pipeline the generated output can be transformed via certain XSL transformations. Finally a serializer changes the temporary SAX stream to the desired output. Figure 4 shows an example of an elevation profile generated for a calculated shortest path between two point of interests.

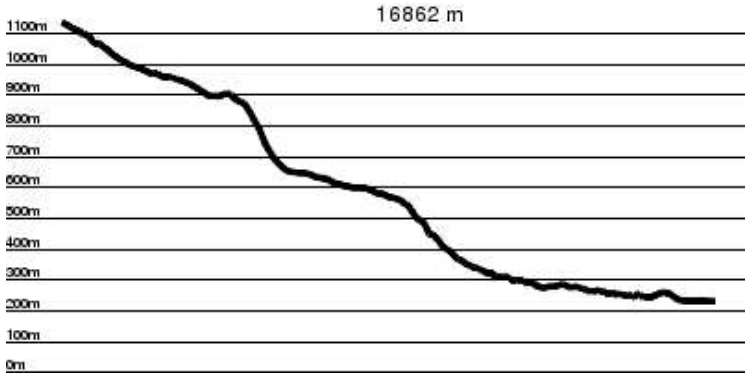


Fig. 4. Generated Elevation Profile

How is this picture generated and integrated into a web page? First of all the user starts a shortest path search between two points of interests (POI). The spatial vector data contains all information about streets and trails for a specific area. The package *OpenJGraph* is used to build an undirected, weighted graph with these information [12]. The result of the Dijkstra algorithm is a set of edges connecting the two POI. These edges are temporarily stored in a database identified by the session key of the user. This result set is taken by the profile generator to generate an SVG-document[3] containing the elevation information of the calculated path. *Cocoon* finally transforms this SVG data into a bitmap file that can be referenced as an image in a web page. This is done by one of the implemented serializers e.g. the `SVG/PNG-Serializer`.

Of course there are other use cases in the depicted web application that integrate spatial information. Here only a few are mentioned:

- Possibility to geocode locations of hotels, restaurants and other point of interests via HTML pages,
- generation of maps showing shortest paths between points of interests,

[3] Scalable Vector Graphics, a W3C recommendation, is a language for describing two-dimensional graphics in XML.

- searching for relevant point of interests within specified distances.

Cocoon can be seen as a mediation between the users spatial interests and different backends described in the next sections.

3.2 Grass

The *Geographic Resources Analysis Support System (Grass)* is one of the biggest and most powerful GIS in the open source world [6]. The program consists of a hundred of modules to process raster, vector and point information. *GRASS* is completely integrated into the MobiHarz architecture but it is not accessed directly during an incoming user request.

It is rather used to generate, administrate and maintain the spatial information used in MobiHarz:

- Combination of different raster layers (e.g. streets, rivers, forests, residential areas) to produce maps of certain needs,
- Digitizing of vector information that cannot be purchased from external sources,
- Interpolation of grid elevation data to raster information in order to build continuous terrain models.

Grass is currently under heavy development. In future versions it will support the already mentioned *PostGIS* implementation from OGC's SQL specification. An improved vector library as well as improved database interfaces will manifest *Grass* as a real alternative to commercial GIS solutions.

3.3 PostgreSQL/PostGIS

PostgreSQL is an object-relational database system that is enhanced by the *PostGIS* project to store and retrieve geographical objects [13]. A *PostGIS*-enabled database is able to store relational attributes as well as spatial properties of objects. Enhanced SQL functions provide a lot of useful methods of analyzing spatial information. In our architecture this database enhancement is used for following tasks:

- Storage of points that have spatial references like bus and train stops, hotels and all POI,
- Storage of digitized line information like streets and hiking trails,
- Temporary storage of calculated shortest paths to enable an automatic generation of relevant images and elevation profiles.

In the near future *PostGIS* integrates some important functionality from the *GEOS* project [5]. Doing this, the implementation comes closer to the mentioned *OpenGIS® Simple Features Specification for SQL*. After that it is planned to commit the implementation for compliance testing to the OGC.

3.4 MapServer

MapServer is responsible for dynamic generation of image maps within the MobiHarz architecture. The program can be used as external library within several programming languages like Perl, PHP, Python and Java. Nevertheless it is mostly used as CGI script. The script offers enhanced functionalities and is configured by simple request parameters [9]. Figure 5 shows the steps that are performed when a user searches for a route between two geographic points.

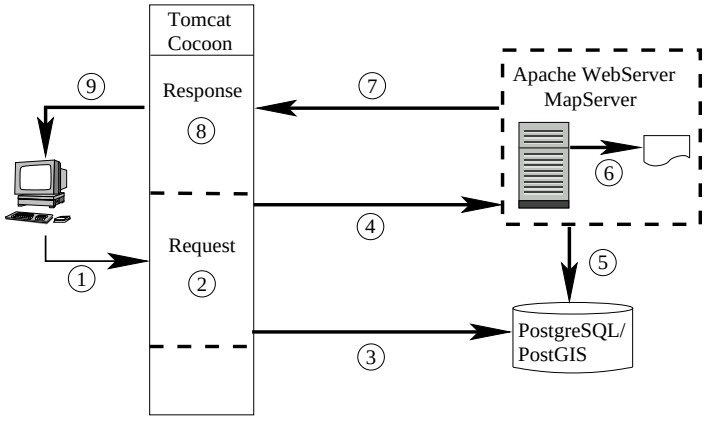


Fig. 5. Shortest Path Search in the MobiHarz System

Following steps are performed to give the user the result to his query:

1. The incoming user request is matched by the responsible pipeline of the *Cocoon* servlet. Start and end point are transferred as HTTP parameters. The user gets a session that identifies him through several HTTP connections.
2. Using these parameters the Dijkstra algorithm of the *OpenJGraph* package performs a shortest path search[4]. Each user query results in a minimum spanning tree for the specified points.
3. The result is used to save the set of lines that connect the two points. The identifying key is an aggregation of the user's session key and current system time.
4. As *MapServer* is running as CGI script it can only be configured via certain HTTP parameters. This is done by the package *HttpUnit*[7]. It establishes a HTTP connection between the server *Cocoon* is deployed on and the web server where *MapServer* is running.

[4] Due to performance issues the graph is built during system startup. It currently consists of 12.000 edges and about 8.000 vertexes.

5. According to the transferred parameters *MapServer* fetches the information about the shortest path from the database. The program is able to read the *PostGIS* format.
6. Additionally it reads further raster images produced by *Grass*. Finally the result image is generated. It can consist of several layers: a normal image map in the background, the shortest path for the query, relevant bus and train stops and further information layers. The image is stored into the file system of the server where *MapServer* is running.
7. As a result *Cocoon* gets an XML document about the generated image. It contains information about the image name, the generated image extent, activated layers and current zoom size of the image.
8. In this step *Cocoon's* pipeline aggregates the different information about the user's search into one XML document.
9. Finally this document is serialized as HTML page and sent to the user. The referenced image map is fetched by the browser through an additional HTTP connection. This request is simply matched by another pipeline of *Cocoon*.

MapServer is able to process a lot of different spatial data formats like `TIFF/GeoTIFF`, `PNG`, `JPEG` and several vector encodings like *Shapefiles* or *PostGIS* information. The product is built on other free software like *FreeType*, *Proj.4* or *libTIFF*.

4 Conclusion

Due to the uncommon structure of spatial information the integration process is different in some ways. Large raster data sets have to be processed within milliseconds, geographical objects must be stored in databases, spatial analysis methods should be available within the application. Especially location based services demand for spatial information about the stored objects.

The architecture depicted in this papers offers almost all of the functionality a spatial web application needs. It consists of some modules that are developed by ourself. The other parts are taken from several open source projects referenced in the paper. Of course there is a lot of other software not described in more detail: *Avalon*, *Castor*, *JDom*, *Jaxen*, *JTidy*, *Batik*.

The modular structure prevents from being dependent on a specific software vendor. Further development is done by thousands of programmers all over the world. As there are no license and update costs this architecture could especially be attractive to public authorities all over the world.

References

1. Norbert Barthelme. *Geoinformatik - Modelle, Strukturen, Funktionen*. Springer Verlag, 3 edition, 2000.
2. Ralf Bill. *Grundlagen der Geo-Informationssysteme - Hardware, Software und Daten - Band 1*. Herbert Wichmann Verlag, Heidelberg, 4 edition, 1999.

3. Apache Cocoon. http://xml.apache.org/cocoon/
4. FreeGis. http://freegis.org.
5. GEOS Geometry Engine Open Source. http://geos.refractions.net/.
6. Official GRASS GIS Homepage - The world Leading Open Source GIS. http://www.geog.uni-hannover.de/grass/.
7. HttpUnit. http://httpunit.sourceforge.net/.
8. Paul Longley, Michael Goodchild, David Maguire, and David Rhind. *Geographic Information Systems and Science*. Wiley-VCH Verlag GmbH, 2001.
9. MapServer Homepage. http://mapserver.gis.umn.edu/index.html.
10. Markus Neteler. *GRASS-Handbuch - Der praktische Leitfaden zum Geographischen Informationssystem GRASS*. Unversität Hannover, 2000.
11. Open GIS Consortium, Inc. (OGC). http://www.opengis.org.
12. OpenJGraph - Java Graph and Graph Drawing Project. http://openjgraph.sourceforge.net/.
13. PostGIS/PostgreSQL. http://postgis.refractions.net.