published in: Proceedings of the 6th Geographic Information Days (GI-Days 2008), Münster, Germany, 2008, 275-296

Interoperable Data Processing by Mobile Geospatial Applications

Thomas Brinkhoff[1], Christof Lindenbeck[2], Jürgen Weitkämper[3]

[1,3]Institute for Applied Photogrammetry and Geoinformatics (IAPG)
FH Oldenburg/Ostfriesland/Wilhelmshaven (University of Applied Sciences)
[1]thomas.brinkhoff@fh-oow.de [3]weitkaemper@fh-oow.de

[2]in medias res, Gesellschaft für Informationstechnologie mbH
Freiburg i. Br.
linde@webgis.de

Abstract. Scarce resources and specific use cases have to be taken into account when building applications for mobile devices. Mobile geospatial applications relying on a web-based spatial data infrastructure (SDI) require special attention. In this paper, we propose a service architecture for integrating mobile clients into an SDI. Important aspects are the computation of adequate maps and geospatial features, the support of offline phases and the supply of context documents that inform about relevant services and datasets. We introduce a prototype of a mobile client that allows using SDI services for retrieval and data acquisition. Finally, a mobile application in tourism using this client is presented.

1 Introduction

Spatial data infrastructures (SDIs) form the backbone of most of today's software applications accessing geospatial information. An SDI provides a framework for spatial data discovery, evaluation, and application. Services of an SDI are typically based on the interface standards developed by the Open Geospatial Consortium (OGC) and the TC 211 of the ISO.

In recent years, geospatial applications are being adapted or transferred to mobile devices. Such mobile applications perform two core tasks in general:

1. As mobile information retrieval system: In this case, the presentation of maps and location-based information forms the center of the application.
2. As data acquisition tool: Here, mobile devices are used to obtain geospatial data in the field. The information can either be entered by a user or taken from sensors connected with the device.

Building software for mobile devices poses severe challenges to the developer – not only in the field of geospatial applications: There exist many different device categories, platforms, and operating systems with diverse user interfaces and performance. Additionally, the life cycle of the underlying technologies is very short, forcing applications to be adapted in short intervals.

Apart from the obvious limitations of mobile devices by scarce resources, a mobile application must address several aspects irrelevant to a corresponding desktop application. Whereas we can assume that a desktop application is always connected to the Internet, a mobile device typically switches between online mode and offline mode very often. This may be due to technical reasons like insufficient signal levels or large energy consumption or simply due to cost. For supporting offline phases, it is important to equip a mobile application with enough information and resources to enable it to work (at least for some period) autonomously. Examples are the use of vector graphics instead of raster maps and of a local database in order to support the temporary storage of modified or new (input or sensor) data. When data acquisition is done, a synchronization may become necessary.

Another problem is that the geospatial data provided by SDIs typically do not fit to the specific requirements of mobile applications. Therefore, a preparation of data is often necessary in order to reduce the complexity of geometries, to deliver data formats that can be efficiently displayed and support offline modes, to reduce the data volume, to simplify the protocols and so on.

In this paper, we propose a service architecture for integrating mobile clients into an SDI that addresses the requirements mentioned before. This architecture allows providing maps and geospatial objects according to the requirements of mobile clients, supporting offline phases and supplying context documents that inform a mobile client about relevant services and datasets. Section 2 of this paper motivates the concept and presents the essential building block of our approach.

In the third section, we introduce a prototype of a mobile client for using SDI services for retrieval and data acquisition. Topics are the visualization, the acquisition and manipulation of data and the communication. The presented concept was inspired by the development of an SDI for disaster management (OKGIS 2008). In addition, we present in section 4 a mobile application in the area of tourism that uses this mobile client. The paper concludes with a short summary and an outlook to future work.

2 Integration of Mobile Clients in a Spatial Data Infrastructure

In this section, we present an architecture for integrating mobile clients into an SDI that addresses the requirements of geospatial applications.

2.1 Providing Adequate Maps and Data

As mentioned in the introduction, mobile clients for geospatial applications are limited by scarce resources. Besides restricted computing power, other important aspects are a relatively small and slow memory, a small display size, special input devices (e.g., a stylus must be supported instead of a mouse or a keyboard), the need to save energy, and varying and often low transmission bandwidths. Furthermore, a large variety of device categories, platforms, and operating systems exists differing in their user interfaces and performance. Therefore, we cannot assume that all popular geospatial web services can be used as by traditional desktop applications. It is necessary to adapt the results of services to the needs of mobile devices. This special treatment can be done

(1) by calling SDI services with parameter sets that fit to the requirements of the mobile client and/or

(2) by modifying the results of SDI services in order to simplify or supersede a post-processing by the mobile client.

An example for the first approach is the use of suitable parameters for the `GetMap` request of a Web Map Service (WMS) (OGC 2006). The objective is to get raster maps as result that fit to the display size and resolution the mobile device that calls the service. In applications like disaster management the specification of the parameters should not be determined by a user himself. Instead, a centralized solution is required that guarantees a certain quality for each of the mobile devices used by the involved institutions.

The second approach, for example, will be performed if the complexity of delivered data does not fit to the capabilities of the mobile device. The complexity refers to the syntax of the data as well as to semantic properties. In case of the Web Feature Service (WFS) (OGC 2005a), the result is an XML-document that describes the geometries by using (the simple feature subset of) the Geography Markup Language (GML). A typical problem for mobile applications is the (unnecessary high) degree of detail of the provided vector geometries. In such a case, a generalization of the geometries is required that takes the display properties of the requesting mobile device into account; the generalization could be performed by another service like the Web Generalization Service presented by Sarjakoski, Sester et al. (2005).

The WFS is a data service delivering features including geometries. However, a client requires graphical objects for displaying purposes. In case of desktop computers, two solutions may be applied: (1) to transform the WFS result into a supported graphic format by the client itself or (2) to use a portrayal WMS that calls the WFS and transforms the XML/GML-document into an image depicting the requested data as map. For mobile devices both solutions are mostly not applicable: For the first approach, the computing power and/or transmission bandwidth are typically too low. The second solution delivers a result that prevents many reasonable autonomous operations to be performed by the mobile client and increases the need for additional communication between client and server (this will be discussed in more detail in the next subsection). The inappropriateness of GML for web-based GIS has been also discussed by Huang et al. (2006).

Our approach for providing mobile applications adequate geospatial data is to introduce an intermediate service called *Mobile Data Service (MDS)*. The MDS is called by the mobile clients for getting individually adapted results from SDI services. For the support of heterogenous mobile devices, a similar approach was followed by Brisboa et al. (2007).

The mobile device can also send `GetMap` requests directly to a WMS with parameter sets provided by the MDS. In the same way, processing services like OpenLS routing or geocoding services can be accessed directly or by using the MDS.

Figure 1 depicts the part of our architecture that provides maps and data. The arrows illustrate the data flow. In order to determine the right parameters for calling SDI services and for processing their results, a database exists that manages profiles for each device type. A more detailed description of the MDS is given by Brinkhoff (2008).

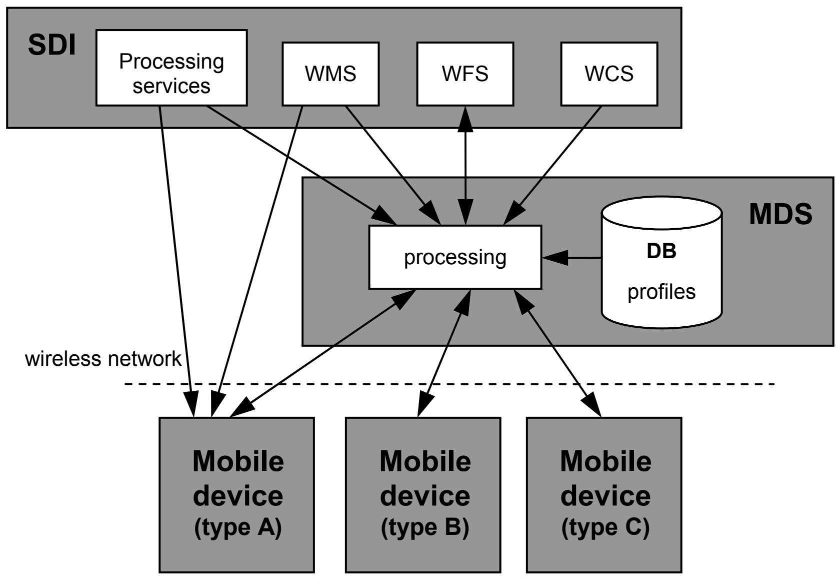

Figure 1: The architecture for integrating mobile applications device-specifically into an SDI (data part)

2.2 Supporting Offline Usage of the Mobile Application

An essential requirement for the mobile application is the ability to continue working in the absence of an Internet connection – so-called *offline mode*. In this section, we discuss several aspects and design considerations that arise from this requirement.

Offline panning and zooming

In the context of an OGC-conformant SDI, each change of viewport, scale or layer visibility results in a new WMS request.

To perform these operations in offline mode, the necessary data has to be present on the device. Thus, some data may be stored on the device in a setup phase of the application. Data that do not change very often, e.g. "background" maps and infrastructure layers, may be pre-installed in this way. For layers containing dynamic data or maps required by unpredicted usage conditions, each time the device is in online mode, the future requirements must be anticipated and the necessary data be requested from the WMS or MDS in advance. For example, a simple strategy would be to enlarge the viewport for a service request resulting from a panning operation.

Offline query processing

In order to query the alphanumeric properties of a geospatial object, the user typically identifies the object with the help of a pointing device.

Let us consider the case of a map that was obtained from a WMS processing data from a WFS. To get the data associated with the object, a `GetFeatureInfo` request must be sent to the WMS indicating the current viewing parameters (viewport, layers etc.) and the pixel coordinates in the image. The WMS transmits the response as structured data, e.g., as an HTML page. In offline mode, we face the problem of determining the object from the pixel coordinates without the help of the WMS. For this purpose, a raster image does not provide sufficient information.

A possible solution to this problem is the usage of a vector representation for the layer data, at least for layers that will be used for interactive object identification (so-called *queryable layers*). According to the WMS standard, a service may deliver vector data, e.g., as *Scalable Vector Graphics* (SVG) (W3C 2003). Many current WMS implementations support this feature. Nevertheless, on closer inspection the SVG documents generated are often ill-suited for our purpose since (among other reasons) they do not provide sufficient information for object identification attached to the SVG display elements.

As a consequence, the vector data representation for a queryable layer must be assembled by the MDS. The MDS issues the WFS request and transforms the XML/GML-document into an SVG document suitable for displaying purposes on the mobile device.

Managing alphanumerical information

In order to work in offline mode, we need the representation of alphanumeric data that belong to the corresponding geospatial feature. This can be done by supplementing the original graphical primitives by additional data sections. For example, SVG allows adding such user-defined elements. However, this approach has an essential disadvantage: First, to embed the additional data in the SVG documents would generate XML files far too large to be processed efficiently on a mobile device. Secondly, the extraction of information from user-defined elements requires querying the XML document, e.g., by XPath expressions which is very costly with respect to computing power. Instead, we favor to provide a separate database containing the corresponding alphanumeric data. Such a database is stored locally on the mobile device and can be queried and updated by the geospatial application. The 1-to-many relation between records in the database and graphical primitives can be represented by using corresponding keys.

The graphical representation and the corresponding database with alphanumerical data can be jointly generated by the MDS and be requested by the mobile client.

Offline editing of data

With the described representation of queryable layers as adapted SVG documents together with an alphanumeric data storage, offline editing of existing objects or object creation can easily be implemented. In the framework of OGC-conformant SDIs such data modifications are submitted to the WFS by `Transaction` requests. Obviously, the transaction request must be postponed until a network connection is present again. This “synchronization” process is described next.

Synchronization and Notification

After a period of offline work, the local dataset in general does not reflect the actual state of datasets organized by the SDI any more. This may be due to user-affected modifications on the device or to alterations of SDI data by some external influence. Thus, when communication with the MDS is re-established after an offline period, the two datasets must be synchronized.

Synchronization works in two directions: data changed on the device has to be incorporated into the SDI by WFS transaction requests and modified data relevant to the device must be transferred to the mobile application. The synchronization is managed by the MDS.

Mobile network connections in general are very slow. Thus, it is essential to decrease the amount of transferred data. In order to avoid resending the same data multiple times to the same recipient, the MDS introduces a “virtual client” for each connected device that keeps track of the state of the data on that device. Whereas up to now the MDS is stateless – much in the spirit of OGC services – due to special requirements induced by mobile devices, a client state managed by the MDS must be introduced.

In principle, there exist two strategies for synchronization. First, the two datasets could be compared to determine which data have to be updated. Secondly, the changes may be logged as they are applied and then later be transmitted to the other dataset.

We favor the “logging” method since far less data have to be exchanged. All changes that the user applies are logged in a local “synchronization” database (see also Figure 5). The synchronization database is transmitted to its virtual representative in the MDS when re-connecting after a period of offline work or periodically in online mode. The MDS can then issue the appropriate WFS transaction request to transmit the changes to the SDI.

When the MDS becomes aware of changes of SDI data relevant to a client, an update notice is stored by the corresponding virtual client. The mobile application may poll the virtual client periodically for these notifications to keep the dataset up-to-date.

2.3 Context Documents for Mobile Applications

Geoapplication Context

The *OGC Web Map Context* (OGC 2005b) has the objective to configure WMS datasets. In a similar – but more extensive – fashion we introduced a so-called *Geoapplication Context (GAC)* for configuring mobile geospatial applications. There are also some similarities of the GAC to the context profile proposed by Predic et al. (2006). A GAC provides information about map services and geospatial data services relevant for the corresponding application, about corresponding data sources, and about suitable processing services. Furthermore, forms are specified that allow the input and/or modification of alphanumerical data. Additionally, input and/or modification operations of geometrical primitives are declared.

The MDS computes the GAC dependent on the device type, the role of its user, the current situation (e.g., the current disaster), and the data already available on the mobile device.

Figure 2 shows a fragment of a GAC that declares a map layer with its corresponding alphanumerical data source. Besides obvious attributes like `id` and `title`, a `Layer` element may contain various attributes controlling the display and interactive behavior. The attributes `minScale` and `maxScale`, e.g., control the visibility of layers for a level-of-detail mechanism like the corresponding parameters in SLD. In case of mobile devices, however, we cannot expect that all layers are available. Therefore, it is reasonable to define a tolerance buffer where the client is free to decide whether to request a new layer or to depict the current layer with some acceptable decrease of quality. The attributes `minScaleHint` and `maxScaleHint` define this tolerance range.

```
<Layer id="LB" title="roads" pickable="0"
    minScaleHint="0.1" maxScaleHint="0.5"
    minScale="0.2" maxScale="0.4"
    dataSource="roaddb"
    .../>
...
<DataSource name="roaddb"
    format     = "sqlite"
    src        = "http://<MDS>/GetData..."
    modifyForm = "mod_road.html"
    newForm    = "new_road.html"
    />
```

Figure 2: Example of a `Layer` element with associated data source

The dataSource attribute of a layer establishes the relation to the corresponding database by referring to the attribute name of a DataSource. As explained in the previous section, some data associated with the layers have to be stored locally on the device. As simplification we assume that the data referring to a layer can be represented by a single database relation which may be a single table or a view referring to one or more tables. The optional attributes modifyForm and newForm refer to HTML pages for editing the alphanumeric data properties of existing and newly defined features, respectively.

Tiling

The GAC offers also information about a suitable tessellation of a layer into tiles. Tiles of a layer can be loaded and unloaded if necessary, which is important for reducing memory consumption. Figure 3 gives an example of a layer tessellated into rectangular tiles.

```
<Layer title="roads" minScale="0.2" maxScale="0.4">
  <Tile src="t10.svg" x="20" y=" 0" width="20" height="20"/>
  <Tile src="t11.svg" x="20" y="20" width="20" height="20"/>
  ...
</Layer>
```

Figure 3: Example of a tiled layer

Processing service declarations

Processing services for a mobile client are declared by the GAC-element Service. Additionally, there may be services provided by the MDS, e.g., for synchronization or polling. Some typical service declarations are depicted in Figure 4. A type attribute is used to distinguish the different services. According to the type of service, further attributes are present. The routing service (type="LS") for example refers to a layer (annLayer attribute) where the calculated routes should be stored.

```
<Service type = "LS"       annLayer="routes"
         src  = "http://<server>/rs" />

<Service type = "GEOCODER" annLayer="routes"
         src  = "http://<server>/rs" />

<Service type = "MDS"      timeInterval="1000"
         src  = "http://<server>/MDS" />
```

Figure 4: Example of Service declarations

2.4 Overall Architecture

Figure 5 illustrates the architecture developed so far. In addition to the part of the architecture depicted in Figure 1, here virtual clients reflecting the current state of individual mobile devices are introduced. A virtual client is instantiated when a mobile device first registers with the MDS and exists until the user logs out of the system.

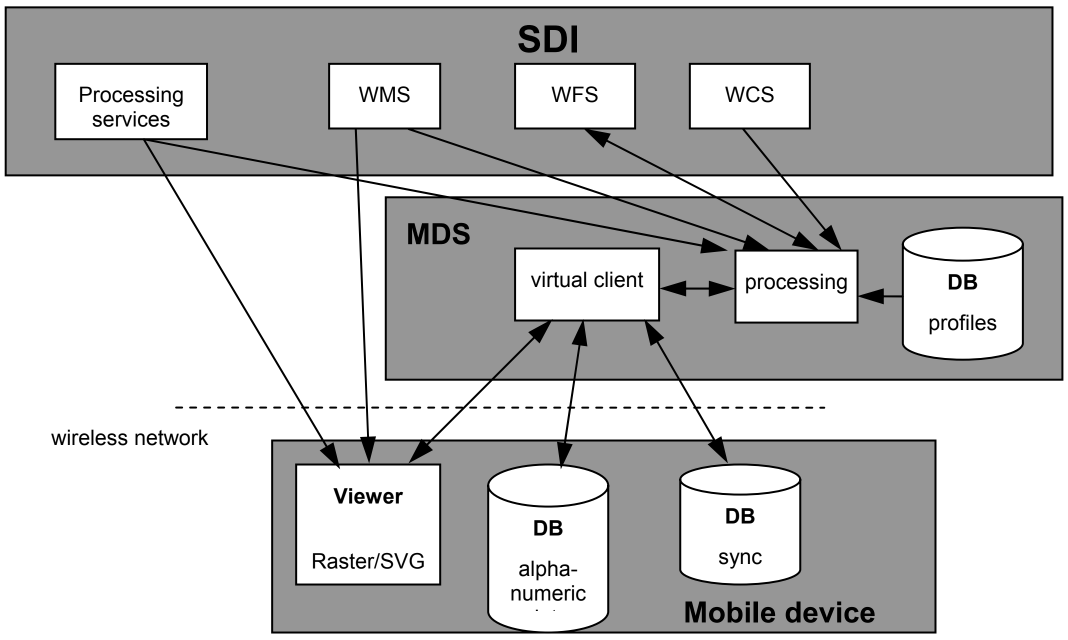

Figure 5: The architecture for integrating mobile devices into an SDI

3 Mobile Client Prototype

In this section, we will present the prototype of the mobile client developed in the context of the project *OKGIS – Open Disaster Management with free GIS-Components* (OKGIS 2008).

The main task of OKGIS is the development and implementation of a concept for the administration, usage, visualization and acquisition of geographic information for disaster management based on open standards. One of the three focal points of OKGIS is the development of a mobile visualization and data acquisition tool supporting staff in the field. The implementation is targeted at mobile devices such as PDAs or TabletPCs and should support sensors attached to the device. With the help of the mobile application, staff in the field may request, view and update geospatial as well as alphanumeric data. Processing services like routing or geocoding services should be accessible directly by the viewer. Additional services specific to disaster management, e.g., support for evacuation scenarios or routing that takes environmental conditions into account, may be accessed via the MDS. The implementation has to address the typical working conditions affecting the mobile device in the field, most important low network bandwidth or the absence of any network connection. Even under these conditions, the mobile application should be seamlessly integrated into the SDI of OKGIS.

The *OKGIS-Viewer* is based on a mobile SVG viewer that originated from a previous project (Brinkhoff & Weistkämper 2005). The prototype is targeted at Windows CE based operating systems like PocketPC, Windows Mobile, and WinCE.NET. A reasonably high performance was achieved by using C++ as implementation language.

The mobile application consists of the following modules: The central visualization component is accompanied by a data store for alphanumeric data, by sensor and communication interfaces as well as by interaction facilities for geometrical and alphanumerical data. Figure 6 depicts the structure of the application. In the sequel, we take a closer look at some of these modules.

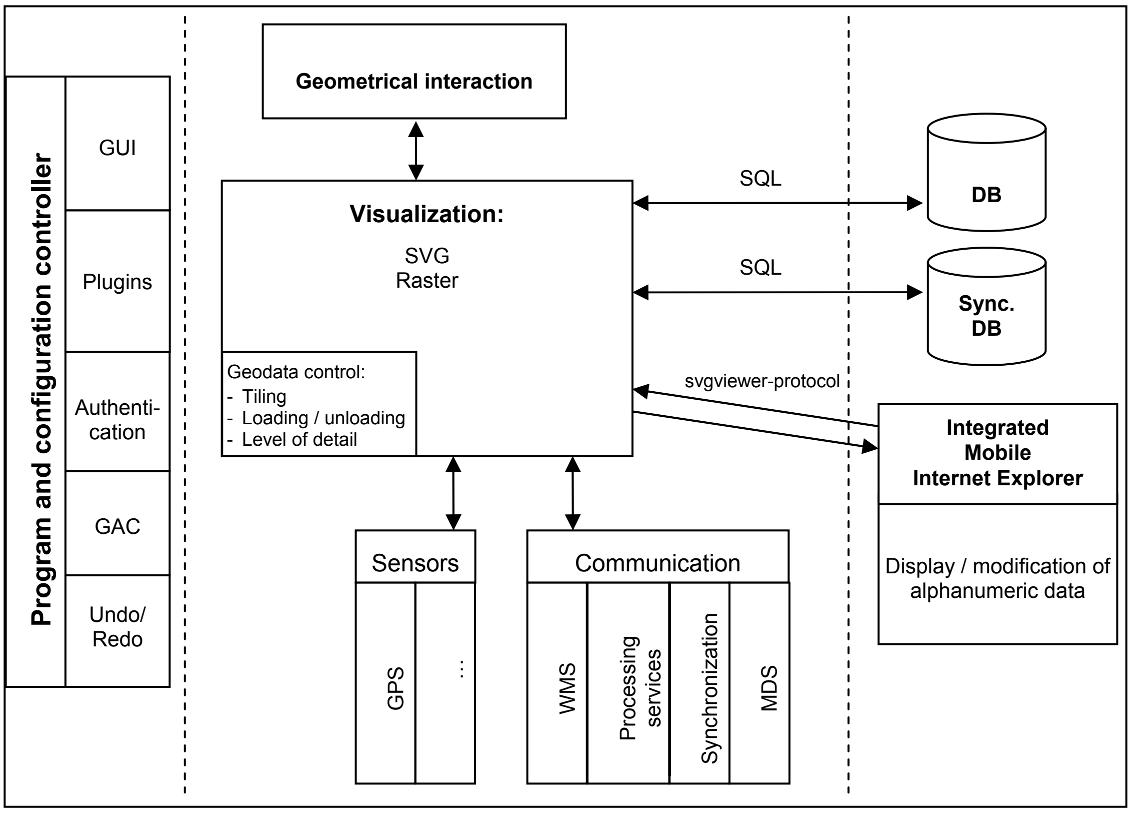

Figure 6: The main components of the prototype of the mobile client

3.1 Program and Configuration Control

The program and configuration controller manages the life cycle of the application.

On program start-up, the layout and functionality of the graphical user interface (GUI) is dynamically set up from the definition supplied in an XML document. Additionally, this document provides information of how to obtain the initial application data. There are two alternatives: Either the GAC is directly referenced by an URL or the address of an authentication service is specified. In the latter case, a GAC is returned from the service after successful login.

The functionality of the mobile client prototype can be extended by plug-ins. These plug-ins are loaded dynamically during program runtime. The GUI configuration document specifies the plug-ins to be used.

3.2 Visualization Component

The data structure of the visualization component is based on the SVG viewer mentioned before. The layers are mapped into an SVG document as depicted in Figure 7. Each layer corresponds to an `image` element of the SVG document. An `image` element of an SVG document represents raster or SVG data. An `image` element either references a data file, a WMS or a MDS `GetData` service.

The SVG viewer issues events to the layer manager when the user interacts with the view, e.g., when panning or zooming. The layer manager is also responsible for loading and unloading of tiles and switching of layer visibility in response to these events.

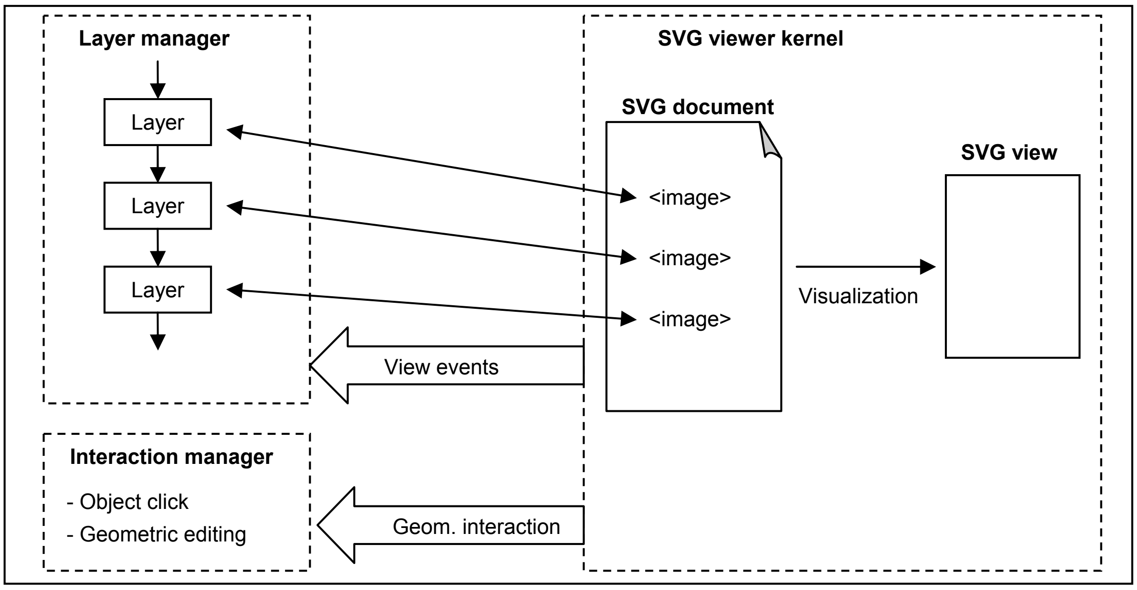

Figure 7: Data representation and visualization by the SVG viewer kernel

Geometric editing operations may be performed by "drawing" on the view with a pointing device like a mouse or stylus. These user interactions are captured by the SVG view and transferred to the interaction manager for further processing.

3.3 Alphanumeric Data Storage

Alphanumeric data are managed on the mobile client by a database management system (see also section 2.2).

Among the many available mobile database implementations SQLite was chosen (SQLite 2008). SQLite shows high performance – even on small devices – and portable to many operating systems. The source code is in the public domain. SQLite is a very compact software library implementing a transactional SQL database engine that is accessible from many different programming languages. Many projects use SQLite as embedded data storage, e.g., *Mozilla Firefox* and *Android* the mobile phone development kit created by Google.

SQLite does not require a separate server process nor is there any installation or configuration necessary. The source code of the database server is linked directly into the program code. SQL statements are executed by simple function calls. The entire database (definitions, tables, indices, and the data itself) is stored as a single file.

When alphanumeric data associated with a layer are requested, the MDS assembles an SQLite database file and transmits it to the client. The same method is used for synchronization. When the mobile client re-connects to the MDS after an offline period, the whole synchronization database file is sent to the MDS.

3.4 Data Acquisition and Manipulation

Alphanumerical data may be displayed and edited by an integrated standard dialog. Alternatively, when more advanced layout or formatting is required, an HTML form may be used (see the `DataSource` declaration in section 2.3). On Windows CE based devices, the *Mobile Internet Explorer* is available as standard web browser.

In general, data from HTML forms are sent to a web server for processing purposes and the web server returns an HTML page as response, which is again displayed by the web browser. The mobile application uses HTML forms in a different way. The form data are directly interpreted to update the database or to change the visual representation of the associated layer. For this purpose, a communication protocol between the web browser and the application was introduced that can be used as a URL protocol with the identifier `svgviewer`. The protocol provides commands to control the visualization component and to access the database.

The implementation uses the Mobile Internet Explorer as ActiveX control that can be directly embedded into the application. The ActiveX control permits the interception of browser events. Thus, when a user activates a link, the URL can be checked for the protocol identifier. URLs using the `svgviewer` protocol are interpreted by the application; all other URLs are submitted to standard processing by the web browser.

3.5 Communication Module

The external service invocations are managed by the communication module of the mobile application.

The mobile application is able to communicate with several standard OGC services like the WMS directly. The current version of the mobile client allows also the direct invocation of OpenLS routing, geocoder and reverse geocoder services.

Additionally, the communication module handles calls to services provided by the MDS. Currently implemented are authentication, notification, synchronization, and data services. The latter transforms WFS data into an SVG document for visualization and an SQLite database file for alphanumeric data handling.

4 APPLICATION IN TOURISM

In addition to the field of disaster management as mentioned in section 3, mobile geospatial applications in other areas can be realized. A prominent example is the field of tourism. For example, a historic city guide using the mobile client was presented by Brinkhoff et al. (2006). A further touristic application is the mobile *Naturpark-Scout* introduced in the sequel.

The development of portable information systems is a rapidly expanding field. The all-encompassing GPS-ization (Schmundt, 2007) is evident due to the activities of global players like Google and Nokia. There are numerous fast evolving open-source projects around the mobile computing of georeferenced data. Most of them are based on Java Virtual Machines like MoWMS (http://www.easywms.com) and Navlet (http://www.navlet.org). The OKGIS-Viewer has been proved as a mobile client technology which solves the requests of the map-centric issues of tourism in the area of nature parks. The modular configuration of data and interface definitions in combination with SVG data allows a wide range of mapping applications for offline outdoor activities.

The Naturpark-Scout (http://www.naturparkscout.de) is online since 2007. It is a web application providing maps of the area around the Town of Bad Wildbad. The projected area is situated in the Black Forest Nature Park Central/North. Tourists can plan their outdoor activities by generating individual hiking/nordic walking (NW) and mountain-biking (MTB) trails. The maps of the area, the generated trails and the points of interest can be exported to the Naturpark-Scout mobile client, which is based on the OK-GIS-Viewer. Visitors of the nature park can use the Naturpark-Scout mobile client as a hiking map, which will be centered at their current position if the device is equipped with a GPS sensor. This project focuses on providing outdoor activity maps that can be used offline without GPRS/UMTS/WLAN connections.

4.1 Architecture of the Naturpark-Scout

The functionality of the Naturpark-Scout is divided into the web application and the mobile client. A typical use case starts by selecting an area of interest in the web application. Individual routing is performed by setting a starting point, points of interest to walk along and an endpoint to finish the route.

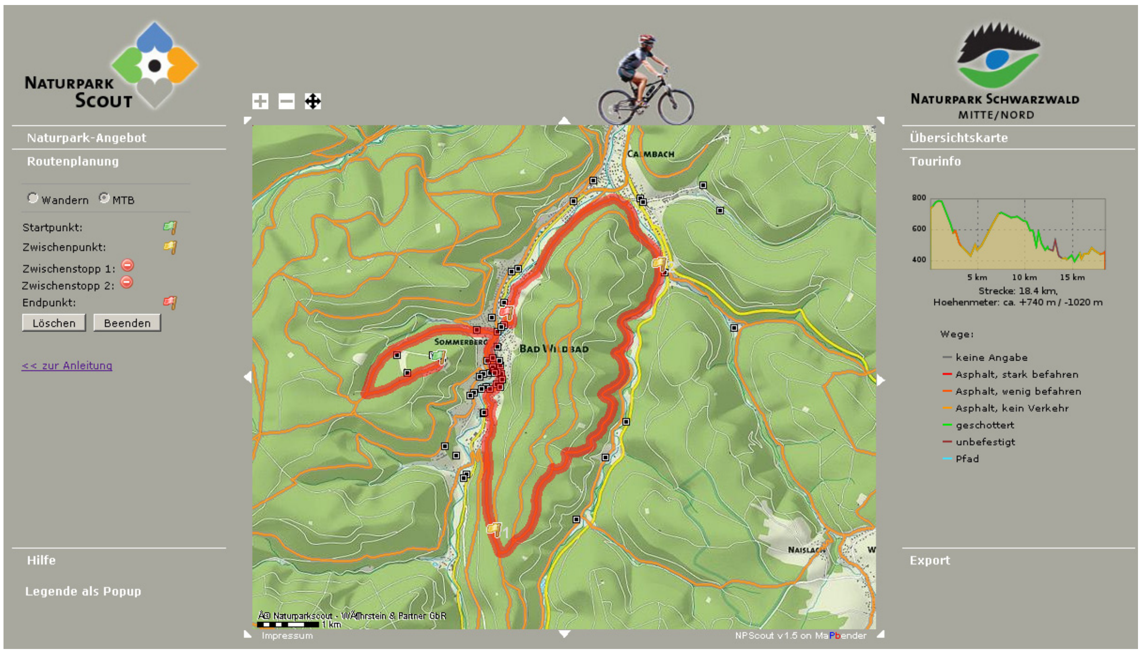

Figure 8: Screenshot of the Naturpark-Scout web interface

The shortest path will be calculated between start and end point. Intermediate points can be set to modify the path.

In order to support decisions, an elevation profile following the route is generated dynamically. The varying conditions of the path are displayed on top of the profile (color-coded). The accumulated altitude is calculated and displayed as the amount to climb uphill and downhill along the route.

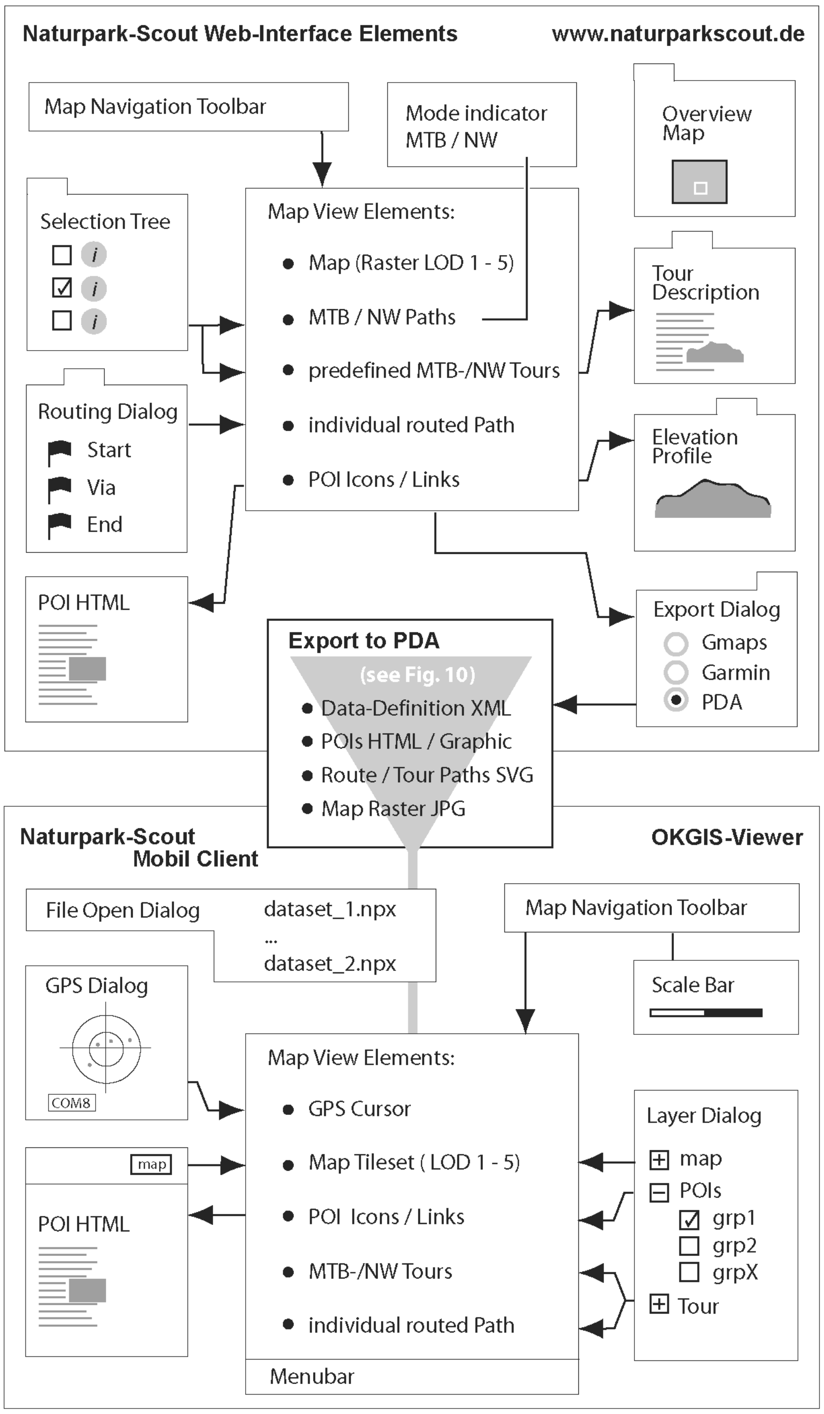

Figure 9: Schematic view of the functional components of the Naturpark-Scout. Elements of the web application (upper part). Symbolic data export (center). Dialogs of the mobile client (lower part).

Figure 9 gives a schematic and user-centered view of the functional components of the Naturpark-Scout system. On the server-side (upper part) different dialogs are used to control the application:

- Navigation controls to pan and zoom in fixed scale increments
- Selection tree for map layer visibility
- Routing dialog to configure individual MTB- or NW-trails
- Overview map (shows the map extend in small scale)
- Tour description of selected predefined trails
- Elevation profile of the last individual routed trail
- Export to target (starts the generation of specific data streams)

Functional components of the mobile client are shown in the lower part of Figure 9. The dialogs are accessible from the menu bar. The file open dialog is used to select a region containing a set of trails and POIs which are generated and stored as discrete data exports. The map navigation uses fixed scale steps to show one of the tiled map sets which are optimized for each level of detail. A scale bar can be activated from the navigation toolbar which can be moved in the map view. The layer dialog is used to control the visibility of trails and groups of POIs. The POI descriptions are coded by HTML. They are displayed in a browser window accessible via POI icons from the map. The browser window contains a map icon for switching back to the map view.

The GPS receiver is activated from the GPS dialog window. The COM port listening for the NMEA data stream can be selected. A graphical control shows the detected satellites. The coordinates of the current position are displayed if a valid position can be processed from the signal.

4.2 Naturpark-Scout Data Export

The raster map, predefined routes and POIs can be downloaded for usage on the mobile device which is symbolized in figure 9 (center). The Naturpark-Scout server processes the area of interest by generating a compressed archive on the fly. It contains the raster data, SVG-, HTML- and image files in a structured folder hierarchy. The content of such an archive is shown in Figure 10.

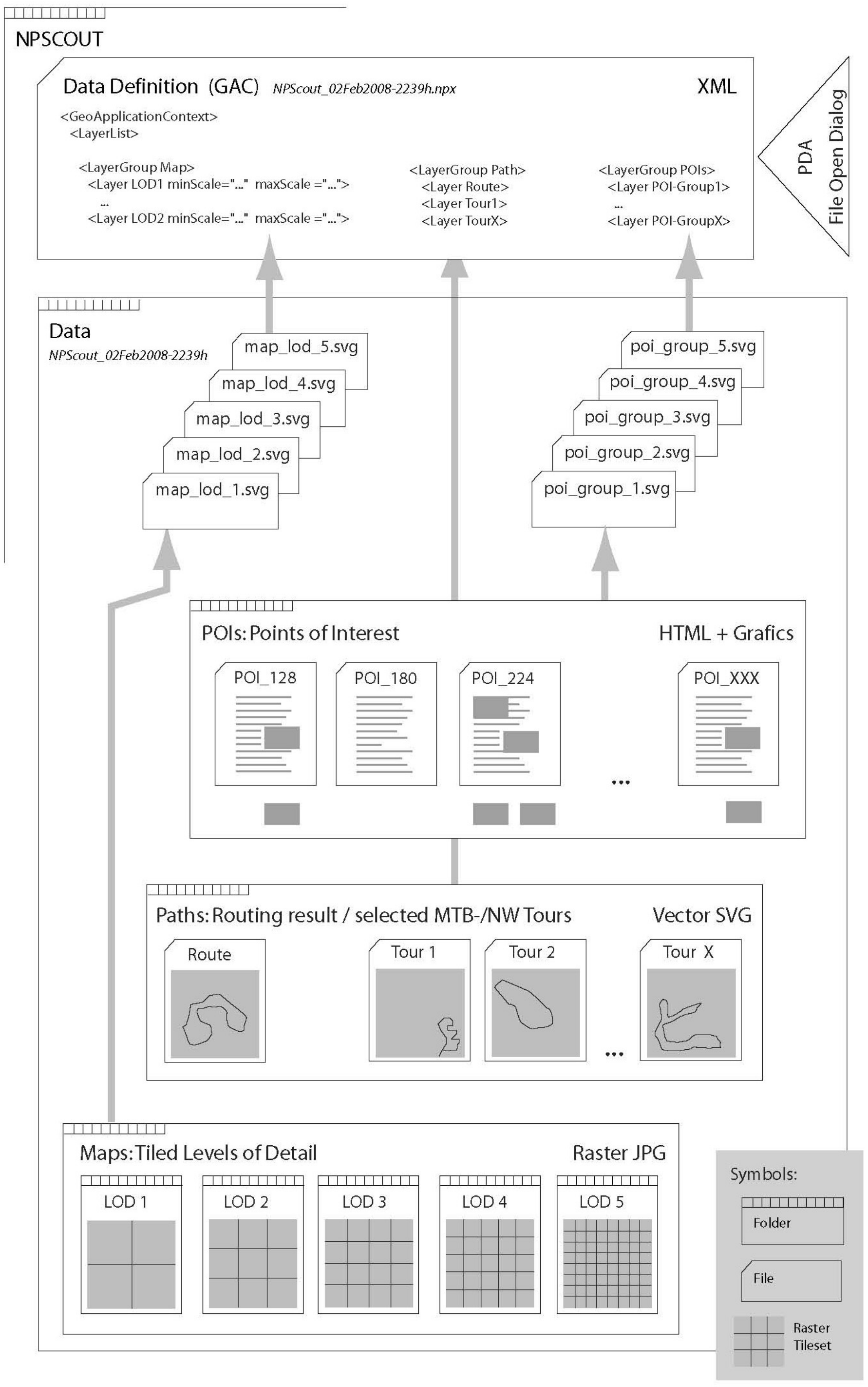

Figure 10: Structure of one discrete OKGIS-Viewer dataset exported from the Naturpark-Scout web application. From top to bottom: Data definition file linking layerlists with data sources. Folder hierarchy containing SVG data and raster images.

The maps are exported in five levels of detail. Every layer is tiled in 400x400 pixel chunks to meet the requirements of the limited PDA memory (Figure 9, bottom part). The branches of raster maps, the vector data of routing results and the grouped POIs are organized in layers. These layers are described in several layer-specific files (Figure 9: `map_lod_X.svg`, `poi_group_X.svg`) using SVG. A central XML-file is used (Figure 9 upper part: data definition) to build the Geoapplication Context (GAC) by referencing all of this layer files and group them into layer lists. This context is used to define additional layer constraints like scale settings. Scale ranges are used to limit the amount of data to be displayed on a single raster map layer at a given zoom level. Predefined fixed scale steps are used to display every LOD raster layer without scaling artifacts.

By uncompressing the downloaded archive, the folder structure will be placed in a time tagged folder (e.g., `NPScout_02Feb2008-2239h`) as root node for the POIs-, paths- and maps-subfolder. The data definition file is stored with a related name (e.g., `NPScout_02Feb2008-2239h.npx`). This time-tagged coding is used to determine different downloads stored on the PDA.

4.3 Naturpark-Scout in the field

The map view is bounded to the current location as long as a valid position can be processed from the GPS-signal. By changing the location, a blinking cursor moves over the map. Approaching the border of the map, the display of the map will be centered again. Depending on the position and the level of detail, different tiles of the map set are displayed. The time used to load and unload map tiles is depending on their size and the speed of the CPU. It is in the range of one to a few seconds. The practical use is restricted by the hardware capabilities: Most of the mobile devices are neither shock- nor waterproof. The displays are small and not bright enough under sunlight conditions. The energy consumption of display and GPS sensors leads to short durations of 1 to 3 hours. Under good weather conditions, the electronic hiking map is a very useful outdoor equipment for unknown areas. In the future, projects like Naturpark-Scout will find wider acceptance due to the on-going development of outdoor PDAs and the widespread usage of wearable devices.

5 CONCLUSIONS

In this paper, we proposed an architecture for integrating mobile geospatial applications into an OGC-compliant SDI. Important aspects supported by this Mobile Data Service (MDS) are the generation of adequate maps and geospatial data, the support of offline phases and the supply of context documents that inform mobile clients about relevant services and datasets. Furthermore, a mobile client – the OKGIS-Viewer – was presented that is based on an SVG viewer and interacts with the MDS or directly with geospatial services. One application of the mobile client – the Naturpark-Scout – was introduced.

The next step will be to integrate the presented architecture into the SDI of OKGIS for supporting disaster management. Further tasks are the use of the MDS for providing further data formats like OGC KML and for accessing SWE (sensor web enabling) services by mobile clients. Finally, further applications should be supported by the MDS as well as by the OKGIS-Viewer.

6 Acknowledgement

Part of this work was done within the project "OK-GIS: Offenes Katastrophenmanagement mit freiem GIS". That project is funded by the German Ministry for Education and Research. This support is gratefully acknowledged.

7 References

Brinkhoff, T. (2008). Supporting Mobile GIS Applications by Geospatial Web Services, accepted for the 21st Congress of the International Society for Photogrammetry and Remote Sensing, Beijing, July 2008.

Brinkhoff, T., A. Gollenstede, P. Lorkowski and J. Weitkämper (2006). "Tourismus und Geoinformatik: Berührungspunkte", Photogrammetrie, Fernerkundung, Geoinformation (PFG), Heft 5/2006, 397-404.

Brinkhoff, T. and J. Weitkämper (2005). "Mobile Viewers based on SVG$^{\pm geo}$ and XFormsGI", Proceedings 8th AGILE Conference on Geographic Information Science, Estoril, Portugal, 2005, 599-604.

Brisboa, N.R., M.R. Luaces, J.R. Parama and J.R. Viqueira (2007). Managing a Geographic Database from Mobile Devices Through OGC Web Services. APWeb/WAIM 2007 International Workshops DBMAN 2007, WebETrends 2007, PAIS 2007 and ASWAN 2007, Huang Shan, China. LNCS 4537, 174-179.

Huang, C.-H., T.-R. Chuang, D.-P. Deng and H.-M. Lee (2006). Efficient GML-native processors for web-based GIS: techniques and tools. Proceedings 14th ACM International Symposium on Advances in Geographic Information Systems, Arlington., VA, 83-90.

OGC (2005a). Web Feature Service (WFS) Implementation Specification, Version 1.1, OpenGIS Implementation Specification, 3 May 2005, http://www.opengeospatial.org/standards/wfs, document 04-094_Web_Feature_Service_Implementation_Specification_V1.1.pdf

OGC (2005b). Web Map Context Documents, Version 1.1.0, OpenGIS Implementation Specification, 19 June 2005, http://www.opengeospatial.org/standards/wms, document 05-005_Web_Map_Context_Documents_WMC_version_1.1.pdf.

OGC (2005c). OpenGIS Location Services (OpenLS): Core Services, 2 May 2005, http://www.opengeospatial.org/standards/olscore, 05-016_OpenLS_core_services_1.1.pdf.

OGC (2006). Web Map Service (WMS) Implementation Specification, Version 1.3.0, OpenGIS Implementation Specification, 15 March 2006, http://www.opengeospatial.org/standards/wms, document 06-042_OpenGIS_Web_Map_Service_WMS_Implementation_Specification.pdf

OGC (2008). Open Geospatial Consortium, http://www.opengeospatial.org.

OKGIS (2008). OKGIS - Open Disaster Management with free GIS-Components, http://www.okgis.de.

Predic B., D. Stojanovic and S. Djordjevic-Kajan (2006). Developing Context Aware Support in Mobile GIS Framework, Proceedings 9th International Conference on Geographic Information Science, Visegrád, Hungrary, 90-97.

Sarjakoski, T., M. Sester et al. (2005). Web Generalisation in GiMoDig – Towards a Standardised Service for Real-Time Generalisation. Proceedings 8th International Conference on Geographic Information Science, Estoril, Portugal, 509-518.

Schmundt, H. (2007). How Pocket Positioning Will Change Daily Life. http://www.spiegel.de/international/spiegel/0,1518,469994,00.html.

SQLite (2008). Software library implementing a SQL database engine. http://www.sqlite.org.

W3C (2003): Scalable Vector Graphics (SVG) 1.1 Specification, W3C Recommendation 14 January 2003, http://www.w3.org/TR/2003/REC-SVG11-20030114.