EXTENSION OF THE OGC WEB FEATURE SERVICE STANDARD FOR MULTIPLE REPRESENTATION DATA

Mark Hampe, Sebastian Intas

Institute of Cartography and Geoinformatics, University of Hannover, Germany - Mark.Hampe@ikg.uni-hannover.de

KEY WORDS: MRDB, Web Service, Cartography

ABSTRACT:

Nowadays improved Internet connections, standardised interchange formats and enhanced mobile devices enable the users to access all information available from almost every place. For the most cases the problem is not the existence of the information but its accessibility, awareness and consistency. Locating the information is mostly coincidental and often detects not the most relevant information. Certain spatial phenomena are represented more than once in different databases, while the content depends on the intention of the data collector. As different applications or situations require alternative information and also visualisations of the data the user should be able to choose between alternative representations and should have access to all information relevant to a certain object of interest. That problem leads to the research field on Multiple Representation Databases (MRDBs), where multiple representations of a certain object are organised in a consistent way in the database. The widespread system architecture for accessing and serving spatial data includes standard interfaces published by the Open Geospatial Consortium (OGC) like Web Feature Service (WFS) or Web Map Service (WMS) and is designed for single representation data including only one single representation per object. In this paper we describe a way to integrate multiple representations data into a web service architecture. We emphasize the need for extending the existing standards to handle multiple representations and propose a system that enables the advantages of an MRDB by extending the OGC WFS.

1. INTRODUCTION

As data collection occurs with different intentions, many information related to a certain real world phenomenon exist side by side without any connection. Even data for different map scales are often not integrated in a common system but exist in parallel. There are several reasons for the existence of multiple representations as well as different types of multiplicity. The reasons for different representations result from different world views (for example a topographer has a different view than a geologist when searching for information), as well as the need for different resolutions of the spatial data in maps. Every user has different requirements regarding the level of detail or content of the data. Bédard & Bernier (2002) name three types of multiplicity of spatial data: *geometric*, *semantic* and *graphic* multiplicity.

In our case we introduce a multiscale database as a special facet of an MRDB. Devogele, Parent & Spaccapietra (1998) note that strictly speaking, the notion of scale (the ratio between the size of an object on the map and its real size on the ground) is a cartographic concept and does not apply to abstract representations stored in spatial databases. The appropriate concepts are: precision (degree of detail in the reporting of a measurement), accuracy (relationship between a measurement and the reality which it purports to represent) and resolution (the smallest object which can be represented) (Goodchild 1991, Müller, Lagrange, Weibel & Salge 1995). In the remainder of this paper the terms multiple resolutions or representations as well as Level of detail (LoD) is used to express this concept.

To organise the data efficiently the multiple representations need to be combined in an MRDB. Such a database combines different representations of the same object in a common system and ensures a consistent storage of the corresponding data by defining links between alternative representations. We gathered multiple LoD of spatial datasets and integrated these data into an MRDB. Section 2. gives an overview of the research work within this area and briefly describes a way to realise an MRDB.

When integrating multiple representational data into a web architecture, problems occur while requesting and handling these data. Standard web architectures and interfaces allow for only one representation per object. The problems resulting from this limitation are described in Section 3.1. To enable the advantages of multiple representations for mobile users the service interfaces have to be modified or extended in a certain way to support such functionalities. A system architecture for a mobile service using standard web-services, the shortcomings of this system when integrating multiple resolution data as well as a proposal to extend the OGC WFS to overcome these drawbacks will be described in Section 3.

Our intention to introduce an MRDB was to use the advantages of this special way of structuring spatial data within a mobile data service. Different map scales are available in a multiple resolutions database, which relieves from the burden of real-time generalisation. The advantages of such a system are manifold: on the one hand, the server can fall back on pre-generalised levels of detail and there is no need for a real-time generalisation; on the other hand, such system allows the user to choose alternative resolutions of the map objects and flexibly zoom in and out. Furthermore the user can access all the information related to a certain phenomenon as the corresponding representations are linked with each other: All information, semantic and geometric, related to a certain phenomenon are available.

2. ORGANISING MULTIREPRESENTATION DATA

As mentioned in the introduction, different facets of multiple representations are thinkable. An MRDB is introduced for effective data maintenance (Bruegger 1996), to facilitate the updating processes of spatial databases (Dunkars 2004) and to allow for multi purpose analyse functionalities in GI Systems (Spaccapietra, Parent & Vangenot 2000).

In our case we concentrate on different representations resulting from cartographic or model generalisation. This leads into a multiple resolution database. Kilpeläinen (1997) describes two requirements of a multiple resolution database:

- The representation levels consist of *different* representations of the same object, which means that geometric representation changes from one level to another.

- Representations at various levels have *connectivities* with each other.

In earlier papers (Hampe, Harrie & Sester 2004, Hampe & Sester 2004) we describe the implementation of an MRDB. The different representation levels are derived by generalising a dataset with a high level of detail. The cartographic generalisation was realised using algorithms for automatic simplification, amalgamation and typification of buildings. Furthermore road data have been simplified through a selection of the road categories as well as line simplification algorithms. Beyond, the possibility to determine corresponding objects through matching algorithms was described.

In the data model the feature types settlement, traffic, water or vegetation are modelled as abstract object classes, whereas each representation is a generalisation (in the sense of Unified Modeling Language (UML)) of its feature type. These subclasses of the feature type class constitute the different representations of a certain object, whereas these representations are linked with each other as they describe the same real world phenomena.

Practically, within a relational database every LoD of a certain feature type is stored as a separate database table. A metadata table lists the names of the tables as well as the feature type these tables are related to and the scale range these representations are applicable best (cf. Figure 1). Metadata are necessary to help applications to select the most appropriate data, like the extended Web Feature Service (eWFS) portrayed in chapter 4.. It describes the object types covered by the database as well as the scale range, covered by each dataset. An extra table describes the

	oid	tablename varchar	scalemin int4	scalemax int4	featuretype varchar
1	110576358	buildings5k	5000	10000	buildings
2	110576359	buildings10k	10000	25000	buildings
3	110576360	buildings25k	25000	50000	buildings
4	110576361	buildings50k	50000	75000	buildings
5	110576362	buildings75k	75000	100000	buildings
6	110576363	buildings100k	100000	200000	buildings
7	110576364	buildings200k	200000	500000	buildings

Figure 1. MRDB metadata

link structure using the IDs of the database objects (cf. Figure 2). Every column of this table stands for a certain level of resolution and describes one link between the levels. Object IDs listed in the same row indicate a connection between these objects. The link between corresponding objects, which is an essential element of an MRDB (Kilpeläinen 1997), reflects that these representations describe the same real world object. The use of mul-

	oid	level1 int4	level2 int4	level3 int4	level4 int4	level5 int4	level6 int4	level7 int4	level8 int4
86	110579882	69	71	18	30	30	20	18	23
87	110579883	90	72	3	26	26	51	18	18
88	110579884	91	73	3	26	26	51	18	18
89	110579885	92	73	3	26	26	51	18	18
90	110579886	93	74	4	26	26	51	18	18
91	110579887	94	76	31	7	7	42	8	8
92	110579888	95	125	31	7	7	42	8	8

Figure 2. Storage of the links in the database

tiresolution data stored consistently in a Database Management System (DBMS) are manifold. The MRDB serves for zooming functionalities, multiresolution maps as well as for accessing all information related to a certain phenomenon. It might be interesting e.g. to get not only attributes linked to a certain building but also information about the district the building is located in or the city. These ideas are described in more detail in (Hampe, Anders & Sester 2003).

3. WEB SERVICES SYSTEM ARCHITECTURE

Within the GiMoDig project (Sarjakoski & Lehto 2003) a prototype serving spatial data from distributed databases for a mobile user has been developed. Lehto (2002) describes the system architecture of this service. This architecture is a typical, standard based framework of a web service providing spatial data. It consists of several separated layers, which are connected through standard based interfaces. The OGC offers a complete set of specifications for setting up a standard based web map service. When software vendors implement their products in compliance with these specifications the user profits from interoperable web based tools for geodata access. The *OGC Web Service Architecture Specification* (OGC 2003) describes a common architectural framework for web based geospatial services. The OGC WFS (OGC 2005*b*) defines interfaces for data access and manipulation operations on geographic features using HTTP as the distributed computing platform. The WMS (OGC 2006) specifies the behaviour of a service that produces georeferenced maps. This standard specifies operations to retrieve a description of the maps offered by a service instance, to retrieve a map, and to query a server about features displayed on a map.

3.1 Web Services for Multiple Resolution Data – Special Requirements

A service providing multirepresentation data should have an interface to access the information easily and efficiently, similar to the OGC WFS. But the OGC WFS or other elements of the OGC Web Services (OWS) (OGC 2003) provide only limited functionalities to support the special requirements of a multiple resolution data service. Problems arise because these interfaces are based on single representations. When multiple representations are introduced, certain functionalities are missing and others will lead into erroneous behaviour of the system. The following examples might clarify this.

When requesting spatial data through a WFS, the user sends a *GetCapabilities* request to explore the content of the database as well as the functionalities of the service. An application accessing this service first collects these information through an XML formatted document sent out by the service as an answer for this request. Beside other sections, this document contains *Operation Metadata*, *Filter capabilities* as well as a *FeatureType list*. In the feature type section, all feature types defined by the specific OGC WFS implementation are listed. Each feature type in turn is represented by a certain database table. Within a multirepresentational environment, more than one database table is associated with a certain feature type. That means either every representation has to be treated as an own feature type or there is the need to extend the data model, introducing child elements of the feature type. The first case would lead to an inconsistent datamodel. Furthermore, when listing all database tables available, the relation between corresponding tables is not expressed. That means to handle multiple representations, child elements of the feature type are needed, describing its representations maintained in the database. For example the feature type ”buildings” might have representations stored in the database tables ”buildings10k”, ”buildings50k” or ”buildings100k”, constituting buildings appropriate for the scales 1:10k, 1:50k and 1:100k, respectively. That means the feature type, e.g. ”buildings”, is just listed in the MRDB metadata table. Only its representations are reflected by database objects owning geometries and attribute data.

To request a certain feature, the OGC WFS offers the *GetFeature* operation. The feature to request can be derived from the list shown in the *GetCapabilites* response document. If every representation would be treated as an own feature type, which would

be the only possibility to access all representations via the OGC WFS, the burden to select the correct representation would be left to the user. Further disadvantages have already been quoted above.

The solution is to follow the strategy of introducing a list of all representations available for a certain feature type. The user or the application just needs to determine a feature type as well as a representation of interest. To be able to select the representation of interest, an additional attribute has to be introduced, a distinctive feature, separating the different representations. For example if the database would maintain different LoD, *scale* would be an attribute to express the difference between the alternative representations (cf. Section 4.). The user might request the feature type ”buildings” and the representation for the *scale* of 1:10k. The distinctive attribute can be different depending on the form of multiplicity. Another MRDB might provide multiple representation in *time* instead of *scale*.

When working with multiple representations the filter functionalities should be extended as well. The extensive filter functionalities described in the *Filter Encoding Implementation Specification* (OGC 2005*a*) offer spatial, comparison, logical, object identifier, arithmetic, function and other operators. The challenge for the user of a multirepresentation service is to filter an appropriate representation of a certain feature. Using the filter operators, the user could request certain objects by using the feature identifier (fid) of those objects. But in a multirepresentational environment when requesting a certain object, more than one representation is available for this object of interest. That means there is the need to not only filter the object of interest but also a certain representation of this object. The connection between the different representations, stored in the link table of the MRDB, can be used by the service when requesting the representation of interest from the database. For example the user requests ”GetFeature featuretype=buildings scale=25000” and filter functionalities are used expressing the interest in the object ”fid=buildings100k.4711” which was selected from the map in use (the name of the *fid* is arbitrarily and does not have to include the scale). The service would select an appropriate representation of the building of interest for the dedicated scale. More details about these functionalities are described in Section 4.

Other filter functions are affected as well. For example geometric filters like bounding box etc. depend on the representation in use. The user should be able to determine the representation to be used for the filtering process. The result of the filtering process would be different, e.g. when using the representations of the scale 1:10k or 1:100k as the objects of the scale 1:10k have a finer granulation.

To enable the full benefits of an MRDB additional request functionalities need to be introduced. The user should be able to request a representation of her choice as described above. But it might also be of interest to get information not only from one representation but additional or all representations available.

Moreover the OGC WFS offers operations to update, delete or insert new data in the database. Also these functionalities have to be adapted because not only one representation is affected through these operations. At the same time this might be a great opportunity using the OGC WFS for automatic update propagations. Using the *DescribeFeature*, *Update*, *Insert* or *Delete* operations would trigger an internal process in the MRDB involving all representations linked to the affected objects.

4. EXTENDED WEB FEATURE SERVICE

To be able to use the advantages of an MRDB by means of a WFS and to overcome the shortcomings described in the previous section, we enhanced the OGC WFS by two operations so far, to result in the extended Web Feature Service (eWFS). The first operation is a slight change of the *GetCapabilites*, whereas *GetAlternativeFeature* was added as a new operation. The new operations are located on top of the existing WFS implementation. Calls not concerning the eWFS will be forwarded to the WFS without any processing. The functions described below can be accessed through a network call using e.g. a web browser.

In the following the architecture of the eWFS is briefly compared to the architecture of the OGC WFS. After that the new operations are introduced. In the examples the differences between the OGC WFS and the eWFS are highlighted with bold letters.

4.1 Architecture

The original architecture (cf. Figure 3) has been extended by another layer. This new layer, which makes up the eWFS, serves as an interface to the outside world. This layer communicates with the WFS and, in addition, with the MRDB (cf. Figure 4). The exchange with the MRDB limits itself, on this occasion, to metadata described in Section 2. No direct access on the WFS and the MRDB is therefore possible from outside.

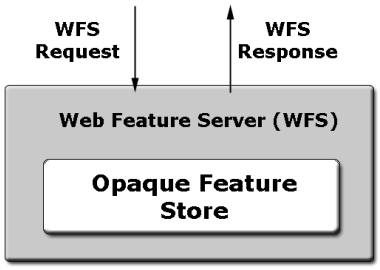

Figure 3. Web Feature Service

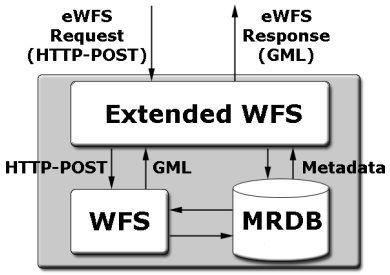

Figure 4. Extended Web Feature Service

4.2 GetCapabilites

The operation *GetCapabilites* is already available from the OGC WFS. Nevertheless, in the extension the response was changed to describe the different representations of a certain feature type, stored in an MRDB. This means that the result of the *GetCapabilites* operation has a slightly changed structure. A major difference is the introduction of a new keyword ”representation” that includes the specification of the name of the representation as well as of metadata like minimum and maximum scale.

The request is expressed as a HTTP-POST:

```
<getCapabilities></getCapabilities>
```

The response to this request coming from an OGC WFS is as follows:

```
<WFS_Capabilities version="1.0.0"
 xsi:schemaLocation="http://
     www.opengis.net/wfs http://...>
  <Service>...</Service>
  <Capability>...</Capability>
  <FeatureTypeList>
    <Operations>...</Operations>
    <FeatureType>
      <Name>...</Name>
      <Title>...</Title>
      <Abstract>...</Abstract>
      <Keywords>...</Keywords>
      <SRS>...</SRS>
      <LatLongBoundingBox .../>
    </FeatureType>
    <FeatureType>...</FeatureType>
    ...
  </FeatureTypeList>
  <ogc:Filter_Capabilities>...</ogc:Filter_Capabilities>
</WFS_Capabilities>
```

The response of the eWFS to *GetCapabilites* request differs to the OGB WFS in the *FeatureType* elements.

```
<WFS_Capabilities version="1.0.0"
 xsi:schemaLocation="http://
     www.opengis.net/wfs http://...>
  <Service>...</Service>
  <Capability>...</Capability>
  <FeatureTypeList>
    <Operations>...</Operations>
    <FeatureType>
      <Name>...</Name>
      <Representation>
        <ReprName>...</ReprName>
        <Scalemin>...</Scalemin>
        <Scalemax>...</Scalemax>
        <Title>...</Title>
        <Abstract>...</Abstract>
        <Keywords>...</Keywords>
        <SRS>...</SRS>
        <LatLongBoundingBox .../>
      </Representation>
      <Representation>...</Representation>
      ...
     </FeatureType>
     <FeatureType>...</FeatureType>
     ...
    </FeatureTypeList>
    <ogc:Filter_Capabilities>...</ogc:Filter_Capabilities>
</WFS_Capabilities>
```

The *FeatureType* in the eWFS describes the abstract real world phenomenon, e.g. settlement or traffic, whereas the term ”representation” reflects its database representation, e.g. ”buildings10k” or ”road50k”. The *FeatureType* node has the name of the layer as the first child node. All other child nodes are *Representation* nodes. I.e. if in the MRDB a layer has five different resolutions, five *Representation* child nodes exist.

A *Representation* node contains further meta information coming from the MRDB, related to this representation.

4.3 GetAlternativeFeature

The *GetAlternativeFeature* operation is available exclusively with the eWFS. It allows to determine certain representations linked to a certain feature type. As in our case the MRDB contains multiple representations covering different scale ranges, this operation serves suitable objects to a given scale. The basic construction is identical with the operation *GetFeature* of the OGC WFS. The *GetAlternativeFeature* operation contains one or several queries (cf. *GetFeature* operation in (OGC 2005*b*)) and furthermore the inclusion of a filter (OGC 2005*a*) is possible. These filter functionalities also have to be adapted. When handling more than one representation the filter can request an arbitrary representation of a feature, as described in the Section 3.

All requests occur as a HTTP-POST. The bold parts correspond exclusively to the eWFS specifications.

Simple request

```
<GetAlternativeFeature>
  <Query typeName=”buildings” scale=”10000”></Query>
</GetAlternativeFeature>
```

This request delivers all data of the feature type ”buildings” which are available for the scale 1:10k. The attribute *scale* specifies the desired representation. If the selected scale does not exist in the MRDB,the resulting data set is empty. It has to be found out, if this solution is suitable for the user or if it would be better to find an alternative representation automatically, displaying the best alternative. Per query only the entry of one feature type and one scale is possible. If the user likes to receive several feature types at the same time, a separate query has to be posed:

```
<GetAlternativeFeature>
  <Query typeName=”buildings” scale=”10000”></Query>
  <Query typeName=”streets” scale=”10000”></Query>
</GetAlternativeFeature>
```

Request with standard filter

```
<GetAlternativeFeature>
  <Query typeName=”buildings” scale=”10000”>
    <Filter>
      <Or>
        <FeatureId fid="buildings10k.0" />
        <FeatureId fid="buildings10k.1" />
        <PropertyIsEqualTo>
          <PropertyName>gid</PropertyName>
          <Literal>17</Literal>
        </PropertyIsEqualTo>
      </Or>
    </Filter>
  </Query>
</GetAlternativeFeature>
```

This request delivers data of the feature type ”buildings”, which are available for the scale 1:10k. The result is filtered, so that only the records 0,1 and 17 are delivered. The filter method indicated here is specified more exactly in (OGC 2005*a*).

Request with new filter element ”ObjectRepresentation”

```
<GetAlternativeFeature>
  <Query typeName=”buildings” scale=”10000”>
    <Filter>
      <Or>
        <ObjectRepresentation>buildings200k.13</ObjectRepresentation>
        <FeatureId fid="buildings10k.5" />
      </Or>
    </Filter>
  </Query>
</GetAlternativeFeature>
```

This request delivers data of the feature type ”buildings” which are available for the scale 1:10k. The filter element FeatureId filters only the element with the id=5, whereas the new filter element *ObjectRepresentation* filters the representations of the destination scale 1:10k which are linked to the object fid=13 in the scale 1:200k.

Request with a standard bounding box

```
<GetAlternativeFeature>
  <Query typeName="buildings" scale="10000">
    <Filter>
      <BBOX>
      ...
      </BBOX>
    </Filter>
  </Query>
</GetAlternativeFeature>
```

This request delivers data of the feature type "buildings" which are available for the scale 1:10k. The result is further filtered, so that only the records within a certain bounding box (BBOX) are delivered (details on BBOX can be found in (OGC 2005*a*)). With the eWFS a modified BBOX is available.

Request with modified bounding box

```
<GetAlternativeFeature>
  <Query typeName="buildings" scale="10000">
    <Filter>
      <Or>
        <BBOX scale='200000'>
        ...
        </BBOX>
      </Or>
    </Filter>
  </Query>
</GetAlternativeFeature>
```

This request also delivers data of the feature type "buildings" which are available for the scale 1:10k. But the resulting dataset is filtered using the BBOX as well as an additional attribute *scale*. That means the bounding box is not applied to the data of the scale 1:10k, but to the data for the scale 1:200k. However, not the records of the resolution 200k are delivered, but the corresponding linked records of the resolution 10k, as desired.

4.4 Extended Transaction WFS

The following description of the Extended Transaction WFS will just reflect some theoretical thoughts at this time, in contrast to the basic service elements implemented and described in the section above. Its implementation will be future work.

The OGC transaction web feature service (WFS-T) would support all the operations of a basic web feature service and in addition it would implement the *Transaction* operation. A transaction request is composed of operations that modify features; that is create, update, and delete operations on geographic features (OGC 2005*b*). The problems of the transaction operations in conjunction with multiple representational data are twofold. On the one hand the operations are based on mono representational data, like the basic operations described above. To handle multi-representational data, an extension is necessary as well. On the other hand the change of data causes inconsistencies in the database, which have to be solved, too. Both facets will be discussed in the following paragraph.

The transaction request is sent via an HTTP-GET where the request is encoded through URL components or HTTP-POST with an XML-formatted document attended to the request. A *Transaction* element may contain zero or more <*Insert*>,<*Update*>, or <*Delete*> elements to create, modify or destroy feature instances.

The <*Insert*> element is used to create new feature instances. The feature to be created has to match the schema described by the *DescribeFeatureType* operation. This operation generates a schema description of feature types serviced by a WFS implementation. The resulting description can be used as a pattern for the insert process. As the schema of a multiple representation database is different to a conventional database, these differences have to be reflected by the feature type description. The function should be able to describe the schema meaning not only the feature type with its attributes but also the whole hierarchy of the representations, similar to the extended *GetCapabilities* function described above.

The *insert* operation itself should be extended by the possibility to name not only the feature type but also the representation the object is meant to be related to. In our case different LoDs are stored for every feature type. When maintaining a multiresolution database, consisting of different LoDs, it is mandatory to allow the user only to change data at the highest level of detail as high LoD data cannot be derived from low LoD data. This would lead to inconsistencies in the database: The features inserted would have no representations in the layers with a higher LoD. Furthermore the user should be able to update not only one representation layer but also insert different representations of a real world object, e.g. a building representation as well as its generalised version. That means a possibility has to be integrated to express that different instances to be inserted represent the same real world object.

The *update* operation can be used to change the property values of certain features in the database. It is possible to update geometry properties as well as attributes of features. To determine the object to be updated within an OGC WFS the feature type of the object is mandatory and furthermore a filter can be used to name the objects which has to be changed. When updating an MRDB the user needs to indicate not only the feature type but also the representation, which leads to the same needs of an extension as in the case of an *insert* operation. Regarding the filter operation the same problems occur here as described above. The user needs to specify which level of detail is affected by the filter request. It should be also possible to apply the filter on all representations of a certain feature type, e.g. if a certain attribute value has to be changed for all representations of a certain feature.

When using the *delete* function also an extension of the OGC WFS is needed to define the representation of the feature type object to be deleted as well as for the filter functionalities. The extension of the filter operations follows the same needs as in the extension of the Basic-WFS.

Furthermore a transaction operation within an MRDB system produces inconsistencies and contradictions in the database, which have to be solved. Contradictions would lead to erratic behaviour as queries for which a user expects to receive the same result, may produce different outcomes. Consistency must preserve the topological, distance, direction, and semantic properties of individual objects, as well as it must preserve the spatial relations between them (Paiva 1998). These problems have been treated by several authors and will not be further discussed here (cf. (Egenhofer, Clementini & Felice 1994, Paiva 1998)). Besides, when updating, deleting or inserting a certain object representation, the geometry needs to be changed also for the homologous representations. Procedures to propagate an update event through all representations in an MRDB are discussed e.g. in (Anders & Bobrich 2004). And finally the data structure needs to be adapted, meaning the links between representations have to be deleted, changed or added.

5. SUMMARY AND OUTLOOK

In this paper we highlighted the problems that occur when integrating an MRDB into a web service architecture based on standardised interfaces. The interfaces deliver comprehensive functionalities to request and provide spatial data. On the other hand when introducing multirepresentational data, problems occur as the specifications are based on a data model containing only single representations. The existing standards do not enable the benefits of multirepresentational data providing more information related to a certain feature as well as alternative geometries. By introducing an additional "representation" layer to the architecture of the OGC WFS it is possible to overcome the disadvantages of the OGC WFS when handling multiresolution data. The new operations allow to access linked objects without addressing the database directly. Furthermore the service facilitates to request an appropriate representation. The user does not need to know the structure of the MRDB. When adding new representations, only the metadata table has to be updated. The metadata table as well as the link table are essential when working with this feature service extension. To complete this exploration, the Transaction Web Feature Service (WFS-T) operations also require an extension when used in combination with an MRDB but on the other hand offer a great opportunity to handle update processes of a whole map series automatically through the web.

The actual research in the field of MRDB is mainly concentrated on the use from the data providers' point of view. But also the data user might be interested in alternative representations of a certain real world object and to select the most appropriate representation. More representations also mean more information related to a certain phenomenon. The user could have access to all these information from different data sources, as all these data are linked to the object of interest.

Our future work will concentrate on further analysing the interfaces of the OGC WFS as well as OGC WMS to find out the needs to extend the functionalities to request multirepresentation data. The OGC WMS will be extended to enable the possibility of multiresolution maps or different map layers of alternative representations. Geometric and topological problems have to be taken into account as well when combining different object resolution in the same map. Moreover the possibilities from the data user's point of view will be further examined.

REFERENCES

Anders, K.-H. & Bobrich, J. (2004), MRDB approach for automatic incremental update, *in* 'ICA Workshop on Generalisation and Multiple Representation, Leicester, 2004'. ICA Workshop on Generalisation and Multiple Representation, Leicester.

Bédard, Y. & Bernier, E. (2002), Supporting Multiple Representations with Spatial View Management and the Concept of "VUEL". Joint Workshop on Multi-Scale Representations of Spatial Data, ISPRS WG IV/3, ICA Commission on Map Generalisation, 7-8 July, Ottawa.

Bruegger, B. P. (1996), Spatial Theory for the Integration of Resolution-Limited Data, PhD thesis, University of Maine.

Devogele, T., Parent, C. & Spaccapietra, S. (1998), 'On spatial database intergration', *International Journal of Geographical Information Systems* 12(3), 335–352.

Dunkars, M. (2004), Multiple representation databases for topographic information, PhD thesis, Royal Institute of Technology (KTH).

Egenhofer, M., Clementini, E. & Felice, P. D. (1994), Evaluating inconsistencies among multiple representations, *in* 'Proceedings of the 6th International Symposium on Spatial Data Handling, Edinburgh, Scotland, UK', pp. 901– 920.

Goodchild, M. (1991), Issue of quality and incertainty, *in* J. C. Müller, ed., 'Advances in Cartography', Pergamon, pp. 113–139.

Hampe, M., Anders, K.-H. & Sester, M. (2003), MRDB Applications for Data Revision and Real-Time Generalisation, *in* 'Proceedings of 21st International Cartographic Conference, Durban/South Africa'.

Hampe, M., Harrie, L. & Sester, M. (2004), Multiple Representation Databases to Support Visualization on Mobile Devices, *in* 'Proceedings of the XXth ISPRS Congress, July12-23, 2004, Istanbul, Turkey, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, XXXV(B4:IV)'.

Hampe, M. & Sester, M. (2004), Generating and using a multirepresentation data-base for mobile application, *in* 'Papers of the ICA Workshop on Generalisation and Multiple Representation, August 20-21, 2004, Leicester'.

Kilpeläinen, T. (1997), Multiple Representation and generalization of geo-databases for topographic maps, PhD thesis, Finish Geodetic Institute.

Lehto, L. (2002), Standards-Based Service Architecture for Mobile Map Applications., *in* 'Proceedings of the 5th AGILE Conference on Geographic Information Science, Palma (Mallorca), Spain, April 25-27, 2002, pp. 501-510.'.

Müller, J., Lagrange, J., Weibel, R. & Salge, F. (1995), Generalization: State of the art and issues, *in* J. Müller, J. Lagrange & R. Weibel, eds, 'GIS and Generalization', Taylor & Francis, pp. 3–17.

OGC (2003), OpenGIS Web Services Architecture, OGC Discussion Paper OGC 03-025, Open Geospatial Consortium.

OGC (2005*a*), Filter Encoding Implementation Specification (Filter), OGC Implementation Specification OGC 04-095, Open Geospatial Consortium.

OGC (2005*b*), Web Feature Service Implementation Secification (WFS), OGC Implementation Specification OGC 04-94, Open Geospatial Consortium.

OGC (2006), Web Map Service (WMS), OGC Implementation Specification OGC 06-042, Open Geospatial Consortium.

Paiva, J. (1998), Topological equivalence and similarity in Multi-Representation Geographic Databases, PhD thesis, University of Maine.

Sarjakoski, T. & Lehto, L. (2003), Mobile map services based on open system architecture, *in* 'Proceedings of the 21st International Cartographic Conference (ICC), Cartographic Renaissance, August 10-16, 2003, Durban, South Africa'.

Spaccapietra, S., Parent, C. & Vangenot, C. (2000), GIS Databases: From Multiscale to MultiResolution, *in* 'Proceedings 4th InternationalSymposium, SARA-2000, Horseshoe Bay, Texas, USA, July 26-29, 2000'.