Development of an OMGeo Open Markup Format for the Open GIS Consortium Web Map Service

320342 - Guided Research Computer Science -

Project Final Report

Alexandru A. Chitea

a.chitea@iu-bremen.de

Project Coordinators:

Dr. Peter Baumann - p.baumann@iu-bremen.de

Dr. Michael Kohlhase - m.kohlhase@iu-bremen.de

International University Bremen April 25, 2005 Spring Semester 2005

Contents

List of Figures

List of Tables

1 Executive Summary

Web Map Services[1] (WMSes) have been successfully used in a broad range of applications, from meteorology to disaster mitigation. Moreover, as academic and research tools, they are employed by geoscientists to explore discrete properties of the Earth's surface, such as, for example, elevation levels or marine shorelines.

As the WMSes in use today are targeted to specific application areas and cultural backgrounds, their application programming interfaces (APIs) and user-interfaces are therefore hard-coded accordingly. However, most of the times geodata maps need to be exchanged between several clients or manipulated by software applications. A good illustration is the interpretation of two national roadmaps for navigation purposes. For example, when using the German Brandenburg WMS[2], the term *Strassenorange*[3] represents a small road, whereas a French WMS might encode the same data with a different keyword. Therefore the combination of visual representation and the geodata itself leads to conflicts[4] between data function and its format, creating bottlenecks in terms of usability.

So far, various schemes for georeferenced data representation have been developed at national scales, but none has yet been adopted as an international standard. The proposed project aimed at separating data appearance from its structure through a semantic-based and context-aware technique [5]. As the OMDoc (*Open Mathematical Documents*) format [1] represents an established method for mathematical knowledge representation and management, we have extended it to encompass geosemantics. Our results indicate the achievement of a higher degree of flexibility in terms of geodata manipulation. Among many others, an envisioned area of application for our project results is the disaster mitigation public services[5]. Moreover, we conclude this research project with an outlook on further possible development.

2 Summary Description

A Web Map Service is an online application that generates maps of georeferenced data. Here, maps are defined to be visual representations of geodata and not the data itself. WMSes employ the digital mapping technique [3] to render the geodata in a pictorial format. This approach implies the superimposing of multiple layers that contain georeferenced data, thus allowing map customization through layer addition or removal[6].

As it can be seen, the digital mapping technique can be abstracted to a modular concept, where the map layers represent the constituent modules. In this case, a clear division should be established between the functional specifications of the data contained by each layer and the layer's style of representation. Since the OMDoc format successfully achieves this task for mathematical documents, we have employed it for the development of a Web Map Service georeferenced data format, OMGeo. As such,

[1] The proposed project refers to the Open GIS Consortium WMS Implementation Specification (version 1.1.1) [3].

[2] The service is restricted to academic use, and hence we cannot provide a valid access URL.

[3] *Strassenorange* stands for orange street in German.

[4] E.g.,the language and abbreviated terms may not be international.

[5] As major natural cataclysms usually span more than one country, it is of paramount importance to handle in a common way the geodata recorded from multiple WMS providers.

[6] Meteorological services use this feature in a real-time manner to monitor updated information.

we have created an OMDoc extension to encompass the semantics of georeferenced data. Given that OMDoc has been created with a modular concept in mind, an extension to this format implied just the creation of several additional *libraries*[7] that describe the georeferenced data. Although OMGeo papers rely on a small set of tag elements that are not supported by the existing OMDoc DTD[8] [2], we have considered the creation of a DTD extension an obsolete feature in the present context[9].

So far, the user-interfaces of the Web Map Services' application programming interfaces (APIs) are hard-coded at the development stage and geared towards a certain application[10]. Therefore, the interaction with a specific WMS is restricted to a fixed set of clients[11]. If OMGeo becomes an Open GIS Consortium[12] standard for georeferenced data, then clients would be able to create OMGeo documents that could be both human and machine understandable. A direct application of such a document could be the dynamic creation of an API's user-interface to meet a specific user's needs.

3 Research Motivation

In this section, we will address the research motivation behind the development of the OMGeo format in a bottom-up approach. First, we will outline the general capabilities of any Web Map Service[13]. Afterwards, following the description of an ideal client-WMS interaction, the current limitations of the applications using the Web Map Services will be inferred with regard to a real-life example.

3.1 WMS Capabilities

A Web Map Service is an online software system that provides georeferenced data maps. As they were defined earlier (see Section 2), maps are the visual representation of geodata and not the data itself. These maps are usually rendered in a pictorial format, and support for different encoding formats[14] is available. Upon requesting a map from a WMS, a client specifies a certain (finite) number of map characteristics. Table 1 outlines these characteristics as they are defined in the Open GIS Consortium WMS Implementation Specification (version 1.1.1) [3].

3.2 Client-WMS Interaction

A standard web browser can ask a Web Map Service to retrieve a map (via a GetMap operation) simply by submitting requests in the form of Uniform Resource Identifiers (URIs) [3]. In addition, when retrieving a map, the client can specify, via

[7]Here, the term *library* replaces *Content Dictionary* (see Section 5.2), which is the official OMDoc term.

[8]DTD stands for *Document Type Definitions.*

[9]The DTD is being replaced by the RelaxNG.

[10]Actually, the language barrier and educational background may also alter the user-WMS interaction.

[11]Here, by *clients* we refer to human users or any software application.

[12]See http://www.opengeospatial.org for more information.

[13]See the Open GIS Consortium WMS Implementation Specification (version 1.1.1) [3].

[14]Most used ones are PNG, GIF and JPEG, and occasionally SVG (Scalable Vector Graphics) or WebCGM (Web Computer Graphics Metafile). [3]

Map feature	Description
Layer(s)	the information to be shown on a map
Style	the style associated with each requested layer
Bounding Box	the portion of the Earth to be mapped; it is axis-parallel
Spatial Reference System (SRS)	the projected or geographic coordinate reference system to be used
Output Format	the pictorial format in which the map will be rendered
Output Size	the width and height of the output in pixels
Background Transparency and Color	the style of the output's pictorial format

Table 1: Open GIS Consortium WMS Capabilities

the WMS user-interface, what and how the information should be shown on the map by selecting different attributes for the parameters described in Table 1.

Furthermore, the individual map layers can be requested from different Web Map Servers, enabling as such the creation of a network of *distributed map servers* [3]. As a direct consequence, a "Cascading Map Server" [3] can be established. This would be a WMS that behaves like a client of other WMSes and at the same time behaves like a WMS to other clients. The "Cascading Map Server" approach represents a useful feature as it can perform additional functions such as output format conversion or coordinate transformation on behalf of other servers.

3.3 Current Limitations

As it was shown in the previous sections, the Open GIS Consortium WMS Implementation Specification (version 1.1.1) provides the general framework for the behavior of a service that produces georeferenced maps. As such, the 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]. Therefore, a Web Map Service is implemented with a high degree of usage flexibility. However, limitations, in terms of client-WMS interaction, are induced at the API and presentation levels. Since currently there is no standard to define how geodata should be modeled, many services develop their online applications and model the geodata to meet specific needs. A common approach in this process is to mix the presentation and data representation levels and abstract them to only one level. This level is afterwards hard-coded into both the API, for data retrieval purposes, and the presentation layer. In order to better illustrate this approach, we will refer to the CubeWerx and Brandenburg WMSes[15] whose user-interfaces can be observed in Figures 1[16] and 2[17].

[15] The two WMSes have been selected as they are conform with the Open GIS Consortium WMS Implementation Specification.

[16] http://www.cubewerx.com/main/demo_centre.html. Retrieved: March 2, 2005.

[17] As the service is restricted to academic use only, it cannot be reached at a specific URL. The caption herewith was taken from a local installation.

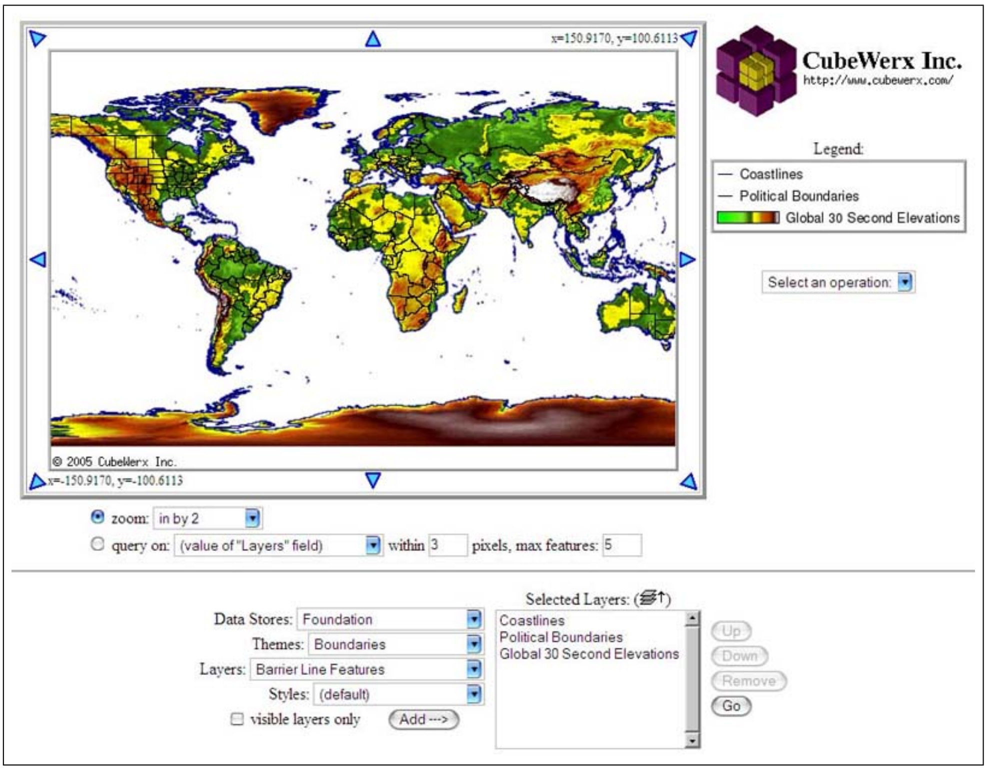

Figure 1: CubeXPLOR Demo by CubeWerx Inc.

Even though at their core the two Web Map Services discussed might provide the same capabilities, their user-interfaces and APIs distinguish them in their functionality. Moreover, this functionality is hard-coded and as such these systems cannot dynamically adapt to different user's needs. As it can be observed from Figures 1 and 2, the first noticeable difference is at the language level. Since neither one of the systems provides any description for its terminology and abbreviations, the applications are targeted towards a fixed group of users.

As we have showed in this section, the strong interdependence between data representation and its format leads to limitations in terms of application utilities. Since the manipulation of georeferenced data retrieved from various WMSes is of paramount importance in application areas like disaster mitigation, preventing the occurrence of data bottlenecks becomes a top priority. As during the past few years, semantic-based and context-aware techniques have pervaded application areas ranging from science and technology to research and education [5], we have envisioned the development of a knowledge representation and management format, OMGeo, as a solution for separating presentation from structure. However, since an established format for mathematical knowledge already existed, OMDoc, which was both semantic-based and context-aware, we have adopted and extend this standard to encompass geosemantics. In Section 4 we will outline the research problems addressed in the thesis herewith.

4 Statement of the Research Problem

As it was shown in the previous sections, the mixture of data format and pre-

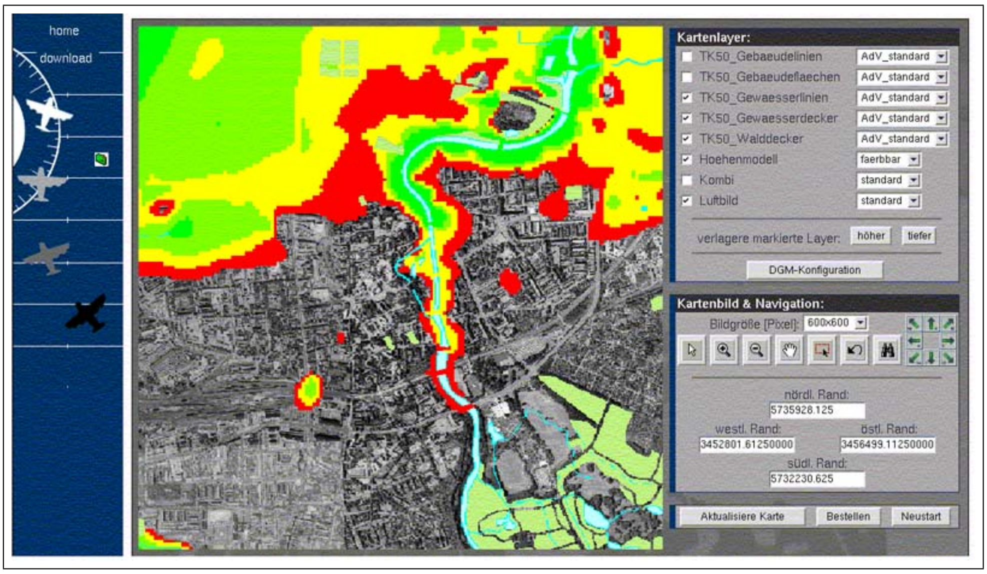

Figure 2: Brandenburg WMS - RasGeo Interface

sentation style considerably reduces the utility of the WMSes. Moreover, since the user-interfaces are hard-coded, clients cannot take full advantage of various WMS applications[18] in the absence of nomenclature standards.

Therefore, the overall research problem referred to the separation of the data representation and presentation levels. Furthermore, a means to encapsulate knowledge about the data in a manner that makes the data content transparent and unambiguous was also researched. More specifically, in the proposed project we have addressed the following research questions:

1. Does OMDoc provide a flexible framework to encompass areas of knowledge, other than mathematics?

2. Can OMDoc be adapted to encompass geosemantics knowledge?

3. How can modularity be achieved so that OMGeo (the extended OMDoc format for georeferenced data) will have an easy way of extending?

As a testing bench for the research claims stated above, the proposed project developed an online demonstrator, that is an independent WMS client that receives OMGeo papers as input and retrieves image data. In Section 5 we will outline the research and development carried out towards the completion of the online demonstrator that can act as a client to various Web Map Services (WMS).

5 Research and Application Development

As it was shown in the previous sections, a map of georeferenced data can be abstracted to a modular concept with the map layers acting as the constituent modules. In order to develop a geosemantics format that overcomes the mixture of appearance

[18] If the user-interface is hard-coded, then the language and the map layers supported are bound to the application in use.

and structure, we have related directly to the aforementioned property. Since another area of knowledge where appearance and structure form two distinct entities is mathematics [4], we have conducted a case-study on whether an extension to an already established semantic-based and context-aware format, OMDoc, was feasible for our research project. The results of this research study laid the basis for the further development. In the following sections the research study, as well as the consequent development are treated in detail.

5.1 OMDoc: A case-study

The OMDoc format is an open markup language for mathematical documents and more generally the knowledge encapsulated in them in a manner that makes their context and content transparent and unambiguous [1]. It approaches this goal by attaching information to mathematical documents that identify the document structure, the meaning of text fragments, and their relation to other mathematical knowledge in a process called *document markup*. As modular design is generally accepted as best practice in the development of any type of complex application, OMDoc format (version 1.2) also features a modularized language approach. To encompass knowledge, OMDoc makes use of a certain type of libraries called *Content Dictionaries* [1]. A Content Dictionary acts as a container for sets of symbol declarations and knowledge about them, and marks them up by **theory** elements. As such, the OMDoc format becomes easily extensible since the provision of knowledge through Content Dictionaries can make the format applicable to other areas, apart from mathematics. Therefore, in order to encapsulate geodata knowledge, the proposed project developed a number of Content Dictionaries that refer to the different constituent blocks of a map. Moreover, a formal mathematical theory (*Maps* theory), which handles the formal description of maps, was derived.

The development of the Content Dictionaries as well as the development of the *Maps* mathematical theory will be outlined in the subsequent sections.

5.2 OMGeo: An OMDoc extension for geosemantics

As a starting point in the development of a WMS client that receives OMDoc documents as input, and produces georeferenced data maps, we have extended the OMDoc format to encompass geosemantical knowledge. This extension was, as already mentioned, two-fold: on the one hand, we have developed a formal mathematical theory dealing with a maps layered-structure, that is the *Maps* theory, and on the other hand we have captured the geosemantical knowledge about map symbols in a formal manner, with the aid of OMDoc Content Dictionaries. Moreover, we have created two **types**[19] within our *Maps* theory. One type was called *omgeo* and the other one *omgeopaper*. Therefore, all elements dealing with the representation of geosemantical knowledge fall within the sort (another denomination of the OMDoc type) *omgeo*, and all the OMGeo papers fall within the *omgeopaper* type.

In the following subsection, we will outline the *Maps* theory and address the geosemantical knowledge captured by the OMGeo Content Dictionaries. Moreover, we will briefly analyze the mark-up structure of an OMGeo paper.

[19] See [1] for more information on what an OMDoc *type* element is.

5.2.1 Maps theory

As OMDoc provides a general framework for handling mathematical theories through its *theory* element, we had to transpose the layered structure of a georeferenced data map to this framework. As such, we have formally developed an OMDoc theory, the *Maps* theory, as a pillar for encompassing geosemantical knowledge in OMDoc. The other important aspect in capturing the geosemantical knowledge was the development of theories for the map symbols, which we have categorized in standardized sets[20].

As it can be seen from line 01 in Figure 3, the map structure was encoded as a mathematical theory. The layered structure of a georeferenced data map has been modeled in the OMDoc format by means of functional constructors.

The *Maps* theory is mainly defined by two functional constructors, encoded as OMDoc symbol elements. These constructors are the following: *composite_map* (line 02 in Figure 3) and *topographic* (line 23 in Figure 3). Moreover, the *maptype* (line 18 in Figure 3) employed by the *composite_map* constructor serves as the return type of a map constructor, e.g. *topographic*.

The *composite_map* constructor, defined between lines 02 and 17 in Figure 3, is an application (indicated by the *OMA*[21] (OpenMath Application tag)) that takes two or more types of maps and associates them together (see lines 11 - 13 in Figure 3). The *topographic* constructor is a function that takes as parameters all the data required by a map layer, and maps it to a *maptype* element. The OMGeo Markup Format defines the following data as required of each constituent map layer: *resolution, srs (spatial reference system), latitude-longitude bounding box, layer bounding box, layer's style, layer's constituent element, layer's attributes.* As it can be seen, some of these attributes are themselves constructors (e.g., see the *resolution* constructor). For a full definition of the *Maps* theory please refer to Section 10 for information on download possibilities.

To conclude this section, we can infer that OMDoc's flexibility in terms of functional constructors helps the modeling of a map's layered structure. However, the desired map elements that need to be rendered pictorially must also be defined in a formal manner, that is as OMDoc theories, and need to be referenced as the *layer's constituent elements* upon map definition. In the following section, we will investigate the capturing of geosemantical knowledge in Content Dictionaries.

5.2.2 Map Elements: Capturing geosemantical knowledge

Another pillar for the development of the OMDoc extension was the development of specific OMGeo *Content Dictionaries*. These act as data libraries that store, in a formal mathematical manner, the geosemantical knowledge for the topographic symbols contained by a map[22].

As the content of the OMGeo *Content Dictionaries* was highly specialized, the research project at hand required a transdisciplinary approach with input from the geoscience field. In order to achieve better content information, we have cooperated

[20] See [6] for more information

[21] See [1] for more information on the OMDoc Markup Format.

[22] In our project we are dealing exclusively with topographic maps, hence the reason why only the *topographic* constructor was defined. However, many other types of maps can be defined in a similar fashion with the *topographic* constructor.

```
...
<theory xml:id="maps">
  <symbol name="composite_map">
    <metadata>
      <dc:description>
        A composite map constructor that takes more maps and composes them together.
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
        <OMA>
          <OMS cd="atypes" name="nassoc"/>
          <OMS cd="maps" name="maptype"/>
          <OMS cd="maps" name="maptype"/>
        </OMA>
      </OMOBJ>
    </type>
  </symbol>
    ...
  <symbol name="maptype">
    <metadata>
      <dc:description>The type of maps,e.g. topographic.</dc:description>
    </metadata>
  </symbol>
    ...
  <symbol name="topographic">
    <metadata>
      <dc:description>
        A topopgraphical map constructor takes the following arguments:
        resolution, srs (spatial reference system), latitudelongitude boundingbox, boundingbox,
        style, elementtype, attributes, and the maptype.
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
        <OMA>
          <OMS cd="simpletypes" name="funtype"/>
          <OMS cd="maps" name="resolutiontype"/>
          <OMS cd="maps" name="srstype"/>
          <OMS cd="maps" name="latlongboundingboxtype"/>
          <OMS cd="maps" name="boundingboxtype"/>
          <OMS cd="maps" name="styletype"/>
          <OMS cd="maps" name="elementtype"/>
          <OMS cd="maps" name="attributestype"/>
          <OMS cd="maps" name="maptype"/>
        </OMA>
      </OMOBJ>
    </type>
  </symbol>
...
```

Figure 3: The *Maps* theory from maps.omdoc

with the *United States Geological Survey (USGS) National Mapping Program*[23]. As such, the map symbols were categorized according to the USGS standards ([6]) and the Content Dictionaries listed in Table 2, that act as geosemantical knowledge containers, have been developed. Aiming to be as exhaustive as possible in our endeavour to provide semantical knowledge for georeferenced data, we have used the 1:24000-scale standard. As such, only a limited set of symbols from the Content Dictionaries stated above can be handled by the developed OMGeo WMS client, since the Web Map Service used by our project deals with the 1:50000-scale standard.

In Figure 4, a map symbol from the *transportation.omdoc* Content Dictionary is presented. In the following we will discuss this symbol with respect to its formal mathematical definition. As stated before, all the map symbols are mathematically defined as theories, having the name of their definition Content Dictionary as an attribute *id*. *Line 01* in Figure 4 opens the transportation theory, that is, the Content Dictionary dealing with symbols representing transportation means. Nested in a the-

[23]See http://www.usgs.gov/ for more information.

Content Dictionary	Description
boundaries.omdoc	symbols used for area delimitation purposes
built_up.omdoc	symbols representing built-up structures
hydrography.omdoc	marine symbols representing structures and water areas
hypsography.omdoc	elevation and contour symbols
named_landforms.omdoc	symbols representing terrestrial structures
nonvegetative.omdoc	symbols representing areas not covered by vegetation
public_land.omdoc	symbols dealing with public building structures
transportation.omdoc	symbols representing transportation means
vegetative.omdoc	symbols representing vegetative surface area

Table 2: Developed Content Dictionaries

```
...
01: <theory xml:id="transportation">
02:   <symbol name="aircraft_facility">
03:     <metadata>
04:       <dc:description>
05:         *An area where aircraft can take-off and land, usually equipped with associated buildings
06:         and facilities.
07:         Delineation: The limit of AIRCRAFT FACILITY is the extent of the area encompassing
08:         all associated structures.
09:       </dc:description>
10:     </metadata>
11:   <type system="omgeo">
12:     <OMOBJ xmlns="http://www.openmath.org/OpenMath">
13:       <OMS cd="maps" name="elementtype" />
14:     </OMOBJ>
15:   </type>
16: </symbol>
...
```

Figure 4: transportation.omdoc

ory tag, there are multiple symbol definitions, like the one between lines 02 and 16 (again please refer to Figure 4). The symbol treated in this case is the *aircraft_facility* map symbol. The symbol's description is found between the *metadata-dc:description* tags. As part of their description, all symbols present a means of delineation. Moreover, the symbol is included in the *omgeo* type (see lines 11 to 15), where it can only act as an *elementtype*, as defined in the Content Dictionary *maps.omdoc* (see line 13).

5.3 OMGeo XML Structure

As outlined in the previous section, the OMDoc format was extended to encompass geosemantical knowledge. As such, the *Content Dictionaries* encompassing geosemantical knowledge along with the *Maps* theory have paved the way for the development of a new type of OMDoc papers: OMGeo mathematical papers. These are papers used to describe georeferenced data maps in a layered-style approach.

However, since we have envisioned the use of OMGeo papers as input to a Web Map Service client, an extra mark-up structure had to be developed on top of the already existing OMDoc one. This mark-up extension deals with WMS-specific parameters, that are required by a WMS client for database retrieval purposes. As it can be seen in Figure 5, the OMGeo mark-up extension includes the following set of tags:

```
<theory xml:id="omgeopaper">
  ...
    <omgeoMap>
        <omgeoURI>http://tlab023.clamv.iu-bremen.de:9000/rasgeo/servlet/rasogc?</omgeoURI>
        <omgeoVersion>1.1.0</omgeoVersion>
        <omgeoService>WMS</omgeoService>
        <omgeoRequest>GetMap</omgeoRequest>
        <mapWidth>400</mapWidth>
        <mapHeight>400</mapHeight>
        <mapFormat>image/jpeg</mapFormat>
        <mapExceptions>application/vnd.ogc.se_inimage</mapExceptions>
        <mapCustomDEM>?</mapCustomDEM>
    </omgeoMap>
  ...
```

Figure 5: The OMGeo XML Structure

- *omgeoMap* - generic tag nesting the OMGeo mark-up extension;
- *omgeoURI* - the URI address where the WMS can be accessed;
- *omgeoVersion* - the version of the service;
- *omgeoService* - the type of service used;
- *omgeoRequest* - the type of WMS request, e.g. *GetCapabilities or GetMap*;
- *mapWidth* - the width of the map (in pixels);
- *mapHeight* - the height of the map (in pixels);
- *mapFormat* - the pictorial format in which the map should be rendered;
- *mapExceptions* - the way exceptions should be handled;
- *mapCustomDEM* - handling elevation levels in a custom-color manner.

5.4 WMS Client Development

As a testing bench for our research project, we have developed an OMGeo WMS client (version 1.0), which is capable of manipulating OMGeo papers to extract geodata images from the Brandenburg Web Map Service. A complete sample of an OMGeo paper is provided in the appendix 10. Moreover, the details of the OMGeo WMS client development follow in the subsequent sections.

6 Development Framework

As the research project developed an OMGeo WMS client to handle OMGeo papers, we outline in this section the software development framework. In particular, we address the environment and tools employed throughout the project's software engineering process.

6.1 Environment and Tools

As the development of the WMS client involved querying a Web Map Service, we have used the Brandenburg WMS. Since this service was restricted to academic use only, it was not publicly available at a specific URL[24]. Therefore, as part of the research project, we have deployed the system on a CLAMV[25] station at International University Bremen. As such, a PostgreSQL[26]/Rasdaman[27] database was installed on top of a Tomcat server. The OMGeo WMS client was developed using the Java 2 Platform, Enterprise Edition[28] (J2EE). In particular, a combination of Java servlets, Java Server Pages (JSP) and Enterprise Java Beans (EJB) technologies was employed. Since the software application was developed under Linux, the Eclipse[29] IDE (Integrated Development Environment) was used.

Moreover, as the client had to parse the OMGeo files to extract relevant data, we have used the Xalan-Java[30] version 2.6.0 API in combination with the Xerces2[31] Java parser as a means of parsing the OMGeo files. The choice was based on the project's functionality needs and moreover, on the widespread employment of the two technologies in other parsing applications.

7 OMGeo WMS Client Description

In this section we will outline the architecture and functionality, in terms of user-perspective, of the OMGeo WMS client. For accessing a complete documentation, generated with javadoc, of all the source code written please refer to Section 10.

7.1 Architecture

The OMGeo WMS client architecture is based on the employment of the technologies outlined in Section 6. The choice of these technologies was made as the Tomcat server, on which the Brandenburg WMS was run, has java support for the *java server pages* technology.

As it can be seen in Figure 6, the OMGeo WMS client architecture has as a core component the *OMGeoParserServlet* class. This servlet receives as input an OMGeo paper file which has been previously uploaded by the user to the application's web server. Upon receiving the file, the servlet dispatches a request to the *OMGeoParser* class. Here, the class methods parse the file using the XPath API provided by the Xalan-j and Xerces2 parsers (see Section 6). The parsed data is afterwards stored in the *OMGeoBean*[32] class. Once the *OMGeoParser* class has finished its operation,

[24] For details on the installation please contact the author of this document or one of the Project Coordinators.

[25] *CLAMV* stands for *Computational Laboratory for Analysis, Modeling, and Visualization.* See http://www.clamv.iu-bremen.de for more information.

[26] See http://www.postgresql.org for more information.

[27] See http://www.rasdaman.com for more information.

[28] J2EE defines the standard for developing component-based multitier enterprise application, featuring Web services support and development tools.

[29] See http://www.eclipse.org for more information.

[30] See http://xml.apache.org/xalan-j/ for more information.

[31] See http://xml.apache.org/xerces2-j/ for more information.

[32] An auxiliary class, *OMGeoLayer*, stores the data pertaining to each individual layer and then this data is passed to the *OMGeoBean* class. However, for the sake of simplicity, we have not represented the interaction between the OMGeoLayer and the OMGeoBean class.

the service is handed back to the *OMGeoParserServlet* class. The servlet is now able to read the parsed data from the *OMGeoBean* class and redirect it to the *omgeo.jsp* web page. The *omgeo.jsp* handles the application's business logic, that is, forming the query for the WMS database, accessing the WMS database and retrieving the map image. As a last action, *omgeo.jsp* outputs the requested information to the user's browser.

In Section 7.2, the interaction described above will become more clear through a client-WMS interaction scenario.

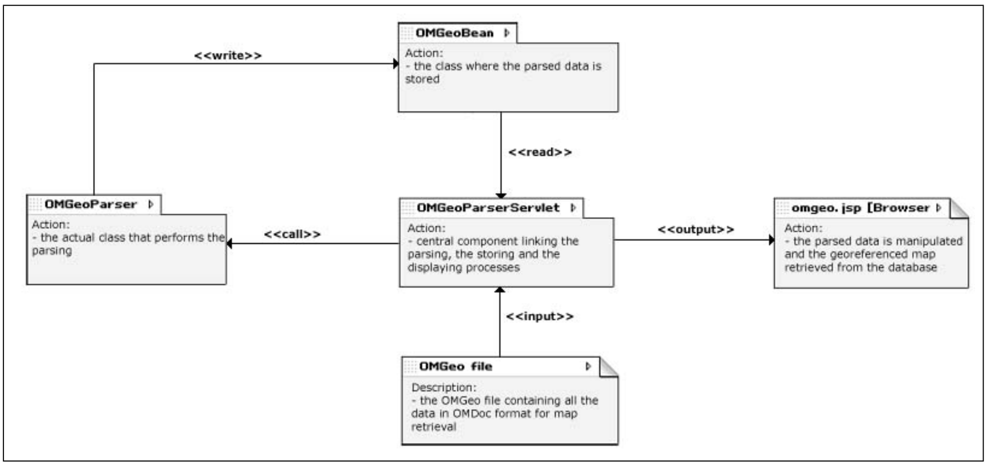

Figure 6: The WMS Client Architecture

7.2 A user's perspective

As the research project at hand regarded the development of a WMS client as a testing bench for our research questions, we have provided a running version (1.0) of this client[33]. The aspect of its Graphical User-Interface (GUI) has been captured in Figures 7 and 8. Moreover, these two figures explain the client-application interaction at GUI level, which can be related to the class interaction described in the previous section.

For the purpose of this demonstrator, we have assumed that the OMGeo paper files are already uploaded on the server-side, and can therefore be accessed from that location by means of a drop-down menu (see Figure 7). Apart from the sample OMGeo papers that contain layered maps, we have included an OMGeo paper that can retrieve a Capabilities file, that is, a file that has the CDATA of the element <*omgeoRequest*> set to *GetCapabilities* instead of *GetMap*.

Upon uploading the file (ses Figure 7)[34], the user is presented with the parsed information, as well as with the extracted georeferenced data map (see Figure 8). When the *Show map* command was issued, the uploaded file followed the path described in Section 7.1. Upon completion of that cycle, the output of the *omgeo.jsp* web page can be seen in Figure 8.

[33] See Section 10 for accessing the OMGeo WMS client.

[34] In this case we are uploading a OMGeo file describing a two-layer map: one showing the trees and the other one representing the water channels, i.e. streams or rivers.

Figure 7: Client - Application Interaction: Client Request

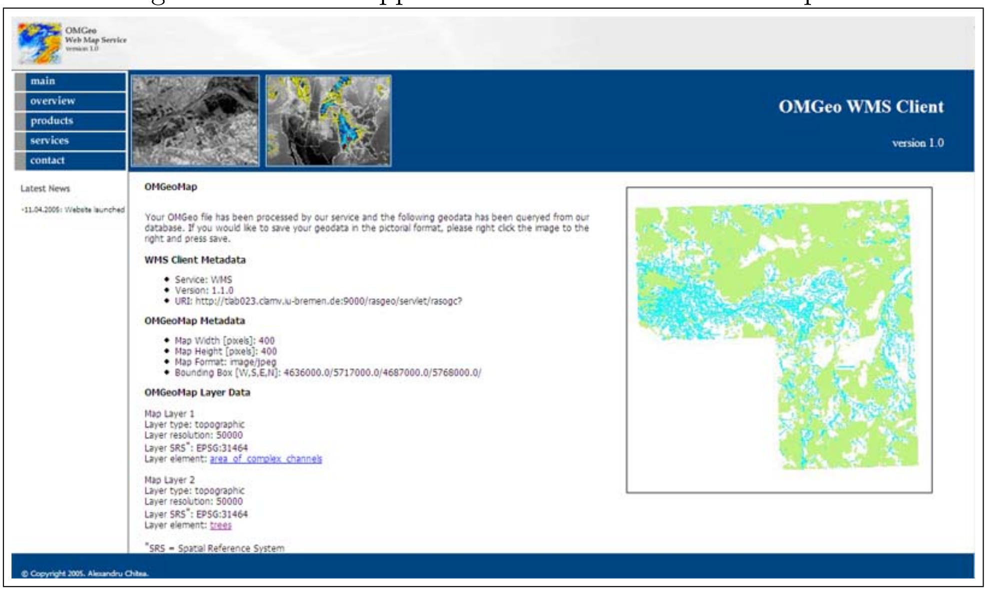

Figure 8: Client - Application Interaction: Service Response

8 Research Conclusions

As a general result, OMDoc provided a flexible framework to encompass areas of knowledge other than mathematics. More specifically, in the research project herewith it has proved capable to capture geosemantical knowledge in mathematical theories. Furthermore, we have managed to employ this knowledge by applying it, in combination with the *Maps* theory, to a client-WMS interaction.

However, a few considerations need to be taken into account when extending the OMDoc format to other areas of knowledge. We have listed them below in their order of importance, and with regard to the newly developed extension, *OMGeo*:

1. The development of the *Maps* theory as a formal mathematical theory was possible because of the maps layered-structure. As such the functional constructors were a perfect fit to modeling the layers of a given map, and finally constructing

the map itself. However, this may not be the case with other areas of semantical knowledge, where functional constructors may prove to be inadequate.

2. Even though logical, the excessive use of constructors can lead to ambiguities in code writing (e.g. the preparation of a 10-layer map with the OMGeo extension). This was primarily the reason why, in the case of the OMGeo extension, we have opted to have a separate element, that is the *<omgeoMap>* tag, to handle the WMS-specific data, instead of handling another set of constructors.

3. Any OMDoc extension should be accompanied by a thoroughly defined schema, and if possible by a specific parser API. The latter should in fact become a requirement if the new extension aims at becoming a standard.[35]

Moreover, the use of OMDoc in the development of a new extension has revealed the need for a new type of constructor: an associative constructor like the one used to compose the different layers of a map. We have to mention that the constructor used in the *Maps* Content Dictionary, that is *nassoc*, is not as yet properly defined, but will be included in a future update of the format.

In Section 9, we conclude this research project with a list of suggestions for further implementations.

9 Outlook and suggestions for further implementations

In this section we will address the functionality enhancements that can improve the future versions of the research project herewith. Moreover, we have also included a list of additional research problems spurred throughout our study of the project.

9.1 Functionality enhancements

In this subsection, we have listed the functional enhancements we believe the OMGeo WMS Client can benefit from in the future:

- Providing a capability to dynamically create the user-interface of the Web Map Service from an OMGeo paper, and the capability to interact with the map.
- Providing a well-developed schema for the OMGeo Markup format as a derivative of the OMDoc one.

9.2 Additional research problems

In this subsection, we have listed the additional research problems that were spurred by the project at hand. They should be considered and implemented, if the OMGeo Markup format becomes a supported standard by the Open Geospatial Consortium.

- The development of an OMGeo parser API.
- Developing the support for *distributed web map servers*, that is, accessing more Web Map Services at a time from the same OMGeo document.

[35] Clients should be capable to easily manipulate through software applications the code that they are writing.

- Developing the support for *cascading web map servers*, that is, when a WMS acts as a client for another WMS, and at the same time as a service for other clients.
- Moreover, the deployment of the OMGeo Markup format at the level where the WMS-database interaction takes place.

10 OMGeo resources and deliverables

In this section we will describe the public resources for working with the OMGeo Markup format and will present the deliverables of the project at hand. However, please note that the OMGeo WMS Client is not publicly available. Access to this service can be granted by the authors only upon written request.

All the other OMGeo resources are available from the MathWeb CVS server at *cvs.mathweb.org*. To check out the deliverables for this project you have to first check out the OMDoc distribution, and then refer to the directory structure for the OMGeo resources listed in Table 3.

directories	**content**
projects/omgis	root directory of the OMGeo project
projects/omgis/cds	the directory for the OMGeo content dictionaries
projects/omgis/docs	the documentation of the project
projects/omgis/examples	ten sample OMGeo papers
projects/omgis/wmsclient	the root directory of the wms-client application

Table 3: OMGeo Resources

To gain access to the CVS repository please contact the authors.

Moreover, for the users of IUB's internal network, the specified resources can be found at the following URL:
http://tlab001.clamv.iu-bremen.de/˜achitea/omdoc/projects/omgis/

APPENDIX A: The *Maps* theory - *maps.omdoc* file

```
<?xml version="1.0" encoding="utf-8"?>

<omdoc xml:id="maps.omdoc"
  xmlns="http://www.mathweb.org/omdoc"
  xmlns:cc="http://creativecommons.org/ns"
  xmlns:dc="http://purl.org/DC"
  xmlns:m="http://www.w3.org/1998/Math/MathML"
  version="1.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.mathweb.org/omdoc
                      http://www.mathweb.org/omdoc/xsd/omdoc.xsd">
  <metadata>
   <dc:date action="created"> 2005-02-17T10:40:50Z </dc:date>
   <dc:title xml:lang="en"> maps </dc:title>
   <dc:creator role="aut"> Alexandru Chitea </dc:creator>
   <dc:source>  </dc:source>
   <dc:date action="updated"> 2005-02-17T10:40:50Z </dc:date>
   <dc:type> Text </dc:type>
   <dc:format> application/omdoc+xml </dc:format>
   <dc:rights>Copyright (c) 2005 Alexandru Chitea</dc:rights>
   <cc:license>
    <cc:permissions reproduction="permitted" distribution="permitted"
     derivative_works="permitted"/>
    <cc:prohibitions commercial_use="permitted"/>
    <cc:requirements notice="required" copyleft="required" attribution="required"/>
   </cc:license>
</metadata>

<catalogue>
  <loc theory="setname1" omdoc="../../../examples/omstd/setname1.omdoc"/>
  <loc theory="simpletypes" omdoc="../../../examples/logics/simpletypes.omdoc"/>
  <loc theory="omtypes" omdoc="../../../examples/omstd/omtypes.omdoc"/>
  <loc theory="built_up" omdoc="built_up.omdoc"/>
  <loc theory="transportation" omdoc="transportation.omdoc"/>
</catalogue>

<theory xml:id="maps">

  <symbol name="composite_map">
    <metadata>
      <dc:description>
A composite map constructor that takes more maps and composes them together.
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
```

```
<OMA>
  <OMS cd="atypes" name="nassoc"/>
  <OMS cd="maps" name="maptype"/>
  <OMS cd="maps" name="maptype"/>
</OMA>
      </OMOBJ>
    </type>
  </symbol>

  <example for="#composite_map">
    <CMP>A simple topographical map:
    <OMOBJ xmlns="http://www.openmath.org/OpenMath">
      <OMA>
<OMS cd="maps" name="composite_map"/>
<OMA>
  <OMS cd="maps" name="topographic" />
  <OMA>
    <OMS name="resolution" cd="maps" />
    <OMI>50000</OMI>
  </OMA>
  <OMA>
    <OMS name="srs" cd="maps" />
    <OMSTR>EPSG:25833</OMSTR>
  </OMA>
  <OMA>
    <OMS name="latlongboundingbox" cd="maps" />
    <OMF dec="51.093867" />
    <OMF dec="10.7151518" />
    <OMF dec="53.6978408" />
    <OMF dec="15.4543864" />
  </OMA>
  <OMA>
    <OMS name="boundingbox" cd="maps" />
    <OMA>
      <OMS name="srs" cd="maps" />
      <OMSTR>EPSG:25833</OMSTR>
    </OMA>
    <OMF dec="3200000.0" />
    <OMF dec="5669000.0" />
    <OMF dec="3530000.0" />
    <OMF dec="5950000.0" />
  </OMA>
  <OMA>
    <OMS name="style" cd="maps" />
    <OMSTR>AdV_Standard</OMSTR>
    <OMSTR>binary</OMSTR>
    <OMSTR>red</OMSTR>
  </OMA>
```

```
  <OMS cd="transportation" name="road"/>
  <OMA>
    <OMS name="attributes" cd="maps" />
    <OMI>0</OMI>
    <OMI>0</OMI>
    <OMI>0</OMI>
  </OMA>
</OMA>
<OMA>
  <OMS cd="maps" name="topographic" />
  <OMA>
    <OMS name="resolution" cd="maps" />
    <OMI>50000</OMI>
  </OMA>
  <OMA>
    <OMS name="srs" cd="maps" />
    <OMSTR>EPSG:25833</OMSTR>
  </OMA>
  <OMA>
    <OMS name="latlongboundingbox" cd="maps" />
    <OMF dec="51.093867" />
    <OMF dec="10.7151518" />
    <OMF dec="53.6978408" />
    <OMF dec="15.4543864" />
  </OMA>
  <OMA>
    <OMS name="boundingbox" cd="maps" />
    <OMA>
      <OMS name="srs" cd="maps" />
      <OMSTR>EPSG:25833</OMSTR>
    </OMA>
    <OMF dec="3200000.0" />
    <OMF dec="5669000.0" />
    <OMF dec="3530000.0" />
    <OMF dec="5950000.0" />
  </OMA>
  <OMA>
    <OMS name="style" cd="maps" />
    <OMSTR>AdV_Standard</OMSTR>
    <OMSTR>binary</OMSTR>
    <OMSTR>red</OMSTR>
  </OMA>
  <OMS cd="transportation" name="road"/>
  <OMA>
    <OMS name="attributes" cd="maps" />
    <OMI>0</OMI>
    <OMI>0</OMI>
    <OMI>0</OMI>
```

```
  </OMA>
</OMA>
      </OMA>
      </OMOBJ>, it has resolution 50000 to 1.
    </CMP>
  </example>

  <symbol name="maptype">
    <metadata>
      <dc:description>The type of maps,e.g. topographic.</dc:description>
    </metadata>
  </symbol>

  <symbol name="topographic">
    <metadata>
      <dc:description>
A topopgraphical map constructor takes the following arguments:
resolution, srs (spatial reference system), latitudelongitude boundingbox,
boundingbox,style, elementtype, attributes, and the maptype.
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
<OMA>
  <OMS cd="simpletypes" name="funtype"/>
  <OMS cd="maps" name="resolutiontype"/>
  <OMS cd="maps" name="srstype"/>
  <OMS cd="maps" name="latlongboundingboxtype"/>
  <OMS cd="maps" name="boundingboxtype"/>
  <OMS cd="maps" name="styletype"/>
  <OMS cd="maps" name="elementtype"/>
  <OMS cd="maps" name="attributestype"/>
  <OMS cd="maps" name="maptype"/>
</OMA>
      </OMOBJ>
    </type>
  </symbol>

  <example for="#topographic">
    <CMP>A simple topographical map:
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
<OMA>
  <OMS cd="maps" name="topographic" />
  <OMA>
    <OMS name="resolution" cd="maps" />
    <OMI>50000</OMI>
  </OMA>
  <OMA>
```

```
    <OMS name="srs" cd="maps" />
    <OMSTR>EPSG:25833</OMSTR>
  </OMA>
  <OMA>
    <OMS name="latlongboundingbox" cd="maps" />
    <OMF dec="51.093867" />
    <OMF dec="10.7151518" />
    <OMF dec="53.6978408" />
    <OMF dec="15.4543864" />
  </OMA>
  <OMA>
    <OMS name="boundingbox" cd="maps" />
    <OMA>
      <OMS name="srs" cd="maps" />
      <OMSTR>EPSG:25833</OMSTR>
    </OMA>
    <OMF dec="3200000.0" />
    <OMF dec="5669000.0" />
    <OMF dec="3530000.0" />
    <OMF dec="5950000.0" />
  </OMA>
  <OMA>
    <OMS name="style" cd="maps" />
    <OMSTR>AdV_Standard</OMSTR>
    <OMSTR>binary</OMSTR>
    <OMSTR>red</OMSTR>
  </OMA>
  <OMS cd="transportation" name="road"/>
  <OMA>
    <OMS name="attributes" cd="maps" />
    <OMI>0</OMI>
    <OMI>0</OMI>
    <OMI>0</OMI>
  </OMA>
</OMA>
      </OMOBJ>, it has resolution 50000 to 1.
    </CMP>
  </example>

  <symbol name="resolutiontype">
    <metadata>
      <dc:description>The type of resolutions of maps</dc:description>
    </metadata>
  </symbol>

  <symbol name="resolution">
    <metadata>
      <dc:description>
```

```
       The  resolution constructor; it takes a number as argument,
       and returns a resolution
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
<OMA>
  <OMS cd="simpletypes" name="funtype"/>
  <OMS cd="setname1" name="N"/>
  <OMS cd="maps" name="resolutiontype"/>
</OMA>
      </OMOBJ>
    </type>
  </symbol>

  <symbol name="srstype">
    <metadata>
      <dc:description>
       The type of Spatial Reference System (SRS) of maps.
      </dc:description>
    </metadata>
  </symbol>

  <symbol name="srs">
    <metadata>
      <dc:description>
       The srs constructor; it takes a string as an argument,
       and returns a srs name.
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
<OMA>
  <OMS cd="simpletypes" name="funtype"/>
  <OMS cd="omtypes" name="string"/>
  <OMS cd="maps" name="srstype"/>
</OMA>
      </OMOBJ>
    </type>
  </symbol>

  <symbol name="latlongboundingboxtype">
    <metadata>
      <dc:description>The latitude longitude bounding box element</dc:description>
    </metadata>
  </symbol>

  <symbol name="latlongboundingbox">
```

```
    <metadata>
      <dc:description>
A latitude longitude bounding box constructor takes the following arguments:
minx, miny, maxx, maxy.
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
<OMA>
  <OMS cd="simpletypes" name="funtype"/>
  <OMS cd="omtypes" name="float"/>
  <OMS cd="omtypes" name="float"/>
  <OMS cd="omtypes" name="float"/>
  <OMS cd="omtypes" name="float"/>
  <OMS cd="maps" name="latlongboundingboxtype"/>
</OMA>
      </OMOBJ>
    </type>
  </symbol>

  <symbol name="boundingboxtype">
    <metadata>
      <dc:description>The bounding box element</dc:description>
    </metadata>
  </symbol>

  <symbol name="boundingbox">
    <metadata>
      <dc:description>
A bounding box constructor takes the following arguments:
SRS constructor, minx, miny, maxx, maxy.
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
<OMA>
  <OMS cd="simpletypes" name="funtype"/>
  <OMS cd="maps" name="srstype"/>
  <OMS cd="omtypes" name="float"/>
  <OMS cd="omtypes" name="float"/>
  <OMS cd="omtypes" name="float"/>
  <OMS cd="omtypes" name="float"/>
  <OMS cd="maps" name="boundingboxtype"/>
</OMA>
      </OMOBJ>
    </type>
  </symbol>
```

```
  <symbol name="styletype">
    <metadata>
      <dc:description>The style associated with an element.</dc:description>
    </metadata>
  </symbol>

  <symbol name="style">
    <metadata>
      <dc:description>
A style constructor takes three string arguments and returns a styletype.
For example, the first argument is usually a standard style, the second is
always black for binary encoding, and the third one is a customly chosen color.
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
<OMA>
  <OMS cd="simpletypes" name="funtype"/>
  <OMS cd="omtypes" name="string"/>
  <OMS cd="omtypes" name="string"/>
  <OMS cd="omtypes" name="string"/>
  <OMS cd="maps" name="styletype"/>
</OMA>
      </OMOBJ>
    </type>
  </symbol>

  <symbol name="elementtype">
    <metadata>
      <dc:description>The element contained by a map.</dc:description>
    </metadata>
  </symbol>

  <symbol name="attributestype">
    <metadata>
      <dc:description>The attributes associated with a map.</dc:description>
    </metadata>
  </symbol>

  <symbol name="attributes">
    <metadata>
      <dc:description>
An attributes constructor takes three natural numbers,i.e. queryable,
opaque and noSubsets and returns a attributestype.
      </dc:description>
    </metadata>
    <type system="omgeo">
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
```

```
<OMA>
  <OMS cd="simpletypes" name="funtype"/>
  <OMS cd="setname1" name="N"/>
  <OMS cd="setname1" name="N"/>
  <OMS cd="setname1" name="N"/>
  <OMS cd="maps" name="attributestype"/>
</OMA>
      </OMOBJ>
    </type>
  </symbol>

</theory>

</omdoc>
```

APPENDIX B: An example of an OMGeo paper

```
<?xml version="1.0" encoding="UTF-8"?>

<omdoc xml:id="omgis.omdoc"
  xmlns:type="http://www.mathweb.org/omdoc"
  xmlns:cc="http://creativecommons.org/ns"
  xmlns:dc="http://purl.org/DC"
  xmlns:m="http://www.w3.org/1998/Math/MathML"
  version="1.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.mathweb.org/omdoc
                      http://www.mathweb.org/omdoc/xsd/omdoc.xsd">

<metadata>
  <dc:date action="created"> 2005-02-17T10:56:35Z </dc:date>
  <dc:title xml:lang="en">OMGeo Sample Paper No. 1</dc:title>
  <dc:creator role="aut"> Alexandru Chitea </dc:creator>
  <dc:source>  </dc:source>
  <dc:date action="updated"> 2005-02-17T10:56:35Z </dc:date>
  <dc:type> Text </dc:type>
  <dc:format> application/omdoc+xml </dc:format>
  <dc:rights>Copyright (c) 2005 Alexandru Chitea</dc:rights>
  <cc:license>
    <cc:permissions reproduction="permitted" distribution="permitted"
     derivative_works="permitted"/>
    <cc:prohibitions commercial_use="permitted"/>
    <cc:requirements notice="required" copyleft="required" attribution="required"/>
  </cc:license>
</metadata>
```

```
<catalogue>
  <loc theory="setname1" omdoc="../../../examples/omstd/setname1.omdoc"/>
  <loc theory="maps" omdoc="../cds/maps.omdoc"/>
  <loc theory="built_up" omdoc="../cds/built_up.omdoc"/>
  <loc theory="vegetative_cover" omdoc="../cds/vegetative_cover.omdoc"/>
</catalogue>

<theory xml:id="omgeopaper">

  <metadata>
    <dc:title>OMGeo Sample Paper No. 1</dc:title>
    <dc:creator>Alexandru Chitea</dc:creator>
    <dc:description>
      This OMGeo paper was written for the Brandenburg Rasgeo WMS Service.
    </dc:description>
  </metadata>

  <omgeoMap>
    <omgeoURI>
     http://tlab023.clamv.iu-bremen.de:9000/rasgeo/servlet/rasogc?
    </omgeoURI>
    <omgeoVersion>1.1.0</omgeoVersion>
    <omgeoService>WMS</omgeoService>
    <omgeoRequest>GetMap</omgeoRequest>
    <mapWidth>400</mapWidth>
    <mapHeight>400</mapHeight>
    <mapFormat>image/jpeg</mapFormat>
    <mapExceptions>application/vnd.ogc.se_inimage</mapExceptions>
    <mapCustomDEM>?</mapCustomDEM>
  </omgeoMap>

  <omtext>
    <CMP>
      This sample OMGeo paper consists of the following map layers:
1. built_up_area
    </CMP>
  </omtext>

  <type system="omgeo">
    <OMOBJ xmlns="http://www.openmath.org/OpenMath">
      <OMA>
<OMS cd="maps" name="composite_map"/>
<OMA>
  <OMS cd="maps" name="topographic" />
  <OMA>
    <OMS name="resolution" cd="maps" />
    <OMI>50000</OMI>
  </OMA>
```

```
  <OMA>
    <OMS name="srs" cd="maps" />
    <OMSTR>EPSG:31464</OMSTR>
  </OMA>
  <OMA>
    <OMS name="latlongboundingbox" cd="maps" />
    <OMF dec="51.093867" />
    <OMF dec="10.7151518" />
    <OMF dec="53.6978408" />
    <OMF dec="15.4543864" />
  </OMA>
  <OMA>
    <OMS name="boundingbox" cd="maps" />
    <OMA>
      <OMS name="srs" cd="maps" />
      <OMSTR>EPSG:31464</OMSTR>
    </OMA>
    <OMF dec="4636000.0" />
    <OMF dec="5717000.0" />
    <OMF dec="4687000.0" />
    <OMF dec="5768000.0" />
  </OMA>
  <OMA>
    <OMS name="style" cd="maps" />
    <OMSTR>AdV_Standard</OMSTR>
    <OMSTR>binary</OMSTR>
    <OMSTR>yellow</OMSTR>
  </OMA>
  <OMS cd="built_up" name="built_up_area"/>
  <OMA>
    <OMS name="attributes" cd="maps" />
    <OMI>0</OMI>
    <OMI>0</OMI>
    <OMI>0</OMI>
  </OMA>
</OMA>
      </OMA>
    </OMOBJ>
  </type>

</theory>

</omdoc>
```

References

[1] Kohlhase, M. (2004). OMDoc - An Open Markup Format for Mathematical Documents (version 1.2) *http://www.mathweb.org/omdoc/omdoc.ps*, manuscript.

[2] Kohlhase, M. (2004). The OMDoc Document Type Definition *http://www.mathweb.org/omdoc/dtd/omdoc.dtd*.

[3] de la Beaujardiere, J. (2002). Web Map Service Implementation Specification (version 1.1.1) *Open GIS Consortium Impplementation Specification*.

[4] Buswell, S. Caprotti, O. Carlisle, D.P. Dewar, M.C. Gaetano, M. Kohlhase, M. (2004). The OpenMath Standard (version 2.0) *The OpenMath Society*.

[5] Kohlhase, M. (2005). Mathematical Knowledge Management Network of Excellence (version 2.0) *Proposal draft*.

[6] USGS Geography: National Mapping Program Standards. Digital Line Graph Standards Documents. *http://geography.usgs.gov/standards/*. Retrieved on: 05/04/2005.