VISUALIZATION OF LANDSCAPE DATA IN DIGITAL MAPS BY EXCLUSIVE USE OF XML-BASED LANGUAGES

Karl Neumann, Peter Ahlbrecht, Silke Eckstein, Brigitte Mathiak and Andreas Kupfer
Technical University Braunschweig, Information Systems Group, P.O. box 3329, D-38023 Braunschweig, Germany, http://www.cs.tu-bs.de/idb, informationssysteme@tu-bs.de

Abstract
The XML-based languages GML, XSLT, and SVG can be used not only for modelling geographic data in an application-oriented fashion, but also for visualizing it in a way closely resembling conventional maps. In this case study, we demonstrate these two features first by remodelling in the Geography Markup Language (GML) real-sized sets of geographic data available from several German authorities for land surveying, and then by mapping this GML-structured data to elements of the language Scalable Vector Graphics (SVG) with the help of the Extensible Stylesheet Language for Transformation (XSLT). In doing so, we model the process of cartographic visualization using XSLT constructs and implement it at the same time. As the result, we obtain graphics which are at least somewhat similar to the corresponding maps provided by those authorities.

INTRODUCTION
The internet is increasingly being used for the exchange of geographic data as well as for computer-based utilization of maps (Kraak and Brown, 2000). Accordingly, also in this context techniques and dedicated languages constructed on top of XML as the carrier language are becoming more popular. The Geography Markup Language (GML), an XML application, is such a language. It has been developed by the Open GIS Consortium as an open standard for the exchange of geographic data (OGC, 2001). Essentially, using the means offered by XML this standard defines a geometry schema and a feature schema which provide a framework for modeling one's own geo-scientific application areas. Another XML-based language, the Extensible Stylesheet Language for Transformation (XSLT), can be used to convert any kind of XML document – and thus also GML documents – into another XML document with a different structure. Mapping geographical objects to graphical objects using XSLT in a way similar to those known from cartographic applications therefore seems obvious. Choosing Scalable Vector Graphics (SVG), another XML-based language, as the target language lies close at hand, as then the entire process of structuring geographic data and supporting its cartographic visualization is realized via XML-based languages. We now demonstrate how this can be done for landscape data and digital topographic maps.

A GML SCHEMA FOR LANDSCAPE DATA
The Geography Markup Language (GML) is a specialized markup language originating from XML and XML Schema (Harold and Means, 2002). GML focuses on geographical features, i.e. descriptions of geographical objects within a frame of reference. The objects' geometrical properties rely on simple, at most two dimensional coordinates; lines between
two given points are always assumed to be straight. The language constructs offered by GML can be used to describe new application areas, meaning most of all that new feature types and the relations between them can be defined.

We focus on the Digital Landscape Model of the German Authorities for Land Surveying as our area of application. Inter alia, this model defines how individual objects of a landscape are constructed, and to which of about 200 given object types they are assigned to. These predefined object types include, for example, housing areas, streets, or lakes, with attributes like road condition, number of roadways, etc. In order to be able to model the Digital Landscape Model with the GML, we introduce a complex type "dlmMemberType", which may contain instances of "dlmMember". As stated by the GML framework, "dlmMemberType" is defined as a feature type. All objects of this class therefore have, among others, a particular geometry and additional attributes. With regard to our application area, the geometrical elements already provided by the GML are well-suited: "location" can be used for points, "centerLineOf" for lines, and "polygonMember" for polygons as shown in the following 15 lines of code:

```
1 <complexType name="dlmMemberType">
2  <complexContent>
3   <extension base="gml:AbstractFeatureType">
4    <sequence>
5     <element ref="dlmOID" minOccurs="0"/>
6     <choice minOccurs="0">
7      <element ref="gml:location"/>
8      <element ref="gml:centerLineOf"/>
9      <element ref="gml:polygonMember"/>
10     </choice>
11     <element ref="Attribute" minOccurs="0"/>
12    </sequence>
13   </extension>
14 </complexContent>
15 </complexType>
```

The additional classes of our GML-based schema are declared in a similar way. The different object types of the Digital Landscape Model, for example "street" or "lake", in our model become subtypes of the "dlmMemberType". Continuing along this structure, instances of the objects can be given, e.g. concrete streets, borders, or administrative districts with their particular coordinates. As a discussion of more advanced features would take up a disproportional amount of space, we refrain from describing them here. Among other things, for instance modelling bridges has not been discussed; they have been realized using references of type "xlink".

**THE XML-BASED LANGUAGE SVG**

In order to visualize GML-encoded data in a way closely resembling conventional maps, we make use of another XML-based language: SVG (Scalable Vector Graphics). SVG describes varying two dimensional graphics in terms of XML (W3C, 2000; Cagle, 2002; Eisenberg, 2002). One of the most important SVG instructions is the "path" statement, which can be used to generate any kind of line or polygon. Figure 1 exemplifies the path statement: it defines a signature for a plant, which is later on used to construct the areal pattern for moorland. Other path statements are required, e.g. for generating the symbols for deciduous trees and coniferous trees, and also the signatures for monuments and towers. Areal patterns are defined using the "pattern" statement. In order to fill a particular area – for instance an area of moorland – with a certain pattern, the corresponding path statement has to be associated with that pattern. We use simple style definitions for this purpose.
As all objects of all object types in their GML-encoded form may be part of a source tree, numerous style definitions are required in order to provide SVG drawing instructions for them. We concentrated on some 50 object types and developed about 70 style definitions and 30 signatures for them. Based on these style definitions and signatures, GML-objects can be mapped onto SVG drawing instructions by means of pattern matching within XSLT templates. As SVG draws in a covering mode, the order in which the complete graphical objects finally are drawn is of importance. We therefore organized the graphical objects into 11 layers, similar to the different layers used in cartographic map construction (Hake et al., 2002; Kraak and Ormeling 2002).

TRANSFORMING GML OBJECTS INTO SVG STATEMENTS VIA XSLT

The XML language XSLT (Extensible Stylesheet Language for Transformation) is a tool for converting XML documents. It resembles a functional programming language and can be used to transform XML documents – and thus also GML documents – into XML documents of a different structure (W3C, 1999; Kay, 2001; Bex et al., 2002). The transformation follows the "pattern matching" principle, i.e. the source tree is analyzed for the occurrence of patterns. These patterns are given in terms of XSLT templates. Additionally, a template also defines how the part of the output tree corresponding to the matching pattern should be constructed.

We implemented each of our GML-represented object classes of the Digital Landscape Model via at least one specialized template. Within a given GML source document, this template searches for the corresponding node type. If the corresponding node is found, e.g. a certain kind of street, then another template is called which, in this case, is responsible for drawing all kinds of lines, for example the line-like street signature. The appearance of a particular line is determined by the suitable style definition, which is passed to the template as a parameter. We have implemented numerous templates – including, e.g., templates for rivers, lakes, forests, or housing areas – each responsible for a certain object class that needs to be represented graphically.

The implemented XSLT templates have been applied to concrete GML objects for landscape data using the freely available "Xalan" XSLT processor (ASF, 2003). For the graphical representation of the final SVG files, conventional web browsers with an SVG viewer plugin can be used. Figure 2 shows a part of the test data – from the Bavarian Authority for Land Surveying – transformed and visualized this way. The entire map generated is 50 by 50 cm in size at a scale of 1:25,000. We processed and visualized several other data sets in addition to this one. As outlined, we only used GML, XSLT, and SVG, but obtained graphics close in quality to professional topographical maps.
In the case study presented in this article, we tried to show that the XML-based language GML is well-suited for modeling geo-scientific application areas, for storing the associated data in a corresponding format, and finally for visualizing them in a way closely resembling conventional maps, all this based on real-sized sets of geographic data. All in all, we hope to have demonstrated that it is possible to realize the entire process of structuring geographic data – and including their cartographic representation – completely on the basis of XML-based languages, and that furthermore all this can be achieved using freely available software and tools only. Aspects like runtime performance and maintainability were not in the focus of our interests. Neither could we compare our GML-conforming schema for objects of the digital landscape model to other schemata, as so far no other alternative GML-schemata are available for this area of application.

The quality of our map-like graphics could be improved by implementing additional templates, style definitions, and pattern signatures, as we so far considered only some 50 object classes. Compared to visualizing additional object classes, however, representing closely adjacent objects is a noticeably more difficult task. It is, for instance, possible that the signature of some avenue trees becomes partly obscured by the line-like signature of a street. Such an occlusion is due to the fact that in maps, trees as well as streets are usually drawn to a much larger scale. Accordingly, here the coordinates of the objects representing avenue trees would have to be transformed with respect to their distance to the geometry of the adjacent street on a case-by-case basis, rather than being processed by a simple transformation (cf. Figure 3). Of course, this advanced process should also take into account that no signatures of even more important objects become obscured. This rather complicated operation of cartographic displacement could, in principle, also be
implemented in XSLT, but a conventional imperative programming language would be better suited.

Figure 3: An example of cartographic displacement.

REFERENCES
Hake, G., Grünreich, D. and Meng, L., 2002: Cartography, 8th ed. (in german), de Gruyter.