AN XML-BASED APPROACH FOR THE COMPOSITION OF MAPS AND CHARTS

Tsoulos, L., Spanaki, M. and Skopeliti, A.

Cartography Laboratory, School of Rural and Surveying Engineering, National Technical University of Athens
9 H. Polytechniou, 157 80 Zographou Campus, Athens, Greece.
Tel: +30 -210-7722730. Fax: +30-210-7722734. E-mail: lysandro@central.ntua.gr, spanaki@mail.ntua.gr
and askop@survey.ntua.gr

ABSTRACT

"Traditional" approaches for the storage and composition of maps/charts on the Web, involve the utilization of commercial geographic/cartographic systems and databases structured and built around a Relational Data model. These approaches impose a number of limitations on the users, the data and the way maps/charts are composed.

The objective of this paper is to describe the way spatial data is stored in an object-oriented environment and subsequently translated in GML format, in order to be used for the composition of maps/charts on the Web. The Geography Markup Language (GML) constitutes a new, powerful means of encoding, transporting, storing and processing spatial data that is compliant with the XML codification and the OGC specification for this data type. The XML data for each fundamental type of spatial geometry (points, lines and polygons) is extracted through spatial queries that operate on the data, which is initially stored in a spatially enabled database.

The XML data can be converted to the GML form, in accordance with three XML Schemas, which have been developed in order to satisfy the application's implementation. Stylesheets appropriate for all three types of spatial geometry have been developed. The conversion of an XML document to its corresponding GML form requires the use of such a stylesheet, which is an XSLT document. The *Extensible Stylesheet Language Transformation* (XSLT) is an XML template supporting the conversion of XML documents from one form to another.

The display of the GML data is accomplished through the SVG's *(Scalable Vector Graphics)* graphic format. The latter is a two-dimensional vector XML template that ensures the visualization of XML data in a web browser environment and overcomes the limitations of traditional bitmap graphics. The XSLT is employed again for the transformation of GML data into SVG graphic output.

Finally, the existence of possible alternative solutions, concerning the drawing of the generated SVG images, is elaborated. The concept of linking the XSL stylesheet of CSS *(Cascading Style Sheets)* files externally has been adopted as the optimal solution, on the grounds of ensuring the independence of the data transformation procedure from the drawing process.

1. INTRODUCTION

For many years, the Standard Generalized Markup Language (SGML), which enables precise document specification, constituted a languished international standard, little used outside few big organizations. Partly this was due to the complexity and the high cost of processing tools. As the web grew, the need for better organizational and searching methods came to the fore. HTML, the language for the web, is in fact an SGML Document Type Definition, and it became apparent that a simpler version of the full SGML standard would be advantageous. Thus XML came about and replaced SGML. The Geography Markup Language (GML) (1) constitutes a new, powerful means of encoding, transporting, storing and processing spatial data that is compliant with the XML codification and the OGC specification for this data type. The XML data for each fundamental type of spatial geometry (points, lines and polygons) is extracted through spatial queries that operate on the data, which is initially stored in a spatially enabled database.

In accordance with the XML Schemas, especially designed to satisfy the application's implementation, the XML data is converted into the GML format. Stylesheets appropriate for all three types of spatial geometry are also developed. The conversion of an XML document to its corresponding GML form requires the use of such a stylesheet, which is an XSLT document. The Extensible Stylesheet Language Transformation (XSLT) is an XML template supporting the conversion of XML documents from one form to another. The display of the GML data is accomplished through the SVG's (Scalable Vector Graphics) graphic format. The latter is a two-dimensional vector XML template that ensures the visualization of XML data and overcomes the limitations of traditional bitmap graphics. The XSLT is employed

again for the transformation of GML data into SVG graphic output. Finally, the existence of possible alternative solutions, concerning the drawing of the generated SVG images, is considered. The road-map for the composition of maps/charts through an XML-based approach is depicted in Figure 1.

2. SPATIAL DATABASE

2.1 Conceptual model

The first step in database design is the creation of the conceptual data model. This model is an abstraction of the real world, which incorporates only those properties thought to be relevant to the application at hand. The data model would normally define specific groups of entities, their attributes and the relationships between these entities. It is independent of the DBMS environment used and its associated data structures (2). In the framework of the research carried out, an application specific spatial model has been developed taking into account the spatial data characteristics and the scope of the spatial data database, which is to store spatial data is an object-oriented environment in order to facilitate subsequent encoding in GML format for the composition of maps/charts. The data model should be compatible with the OGC abstract model of geography (3) used in GML.

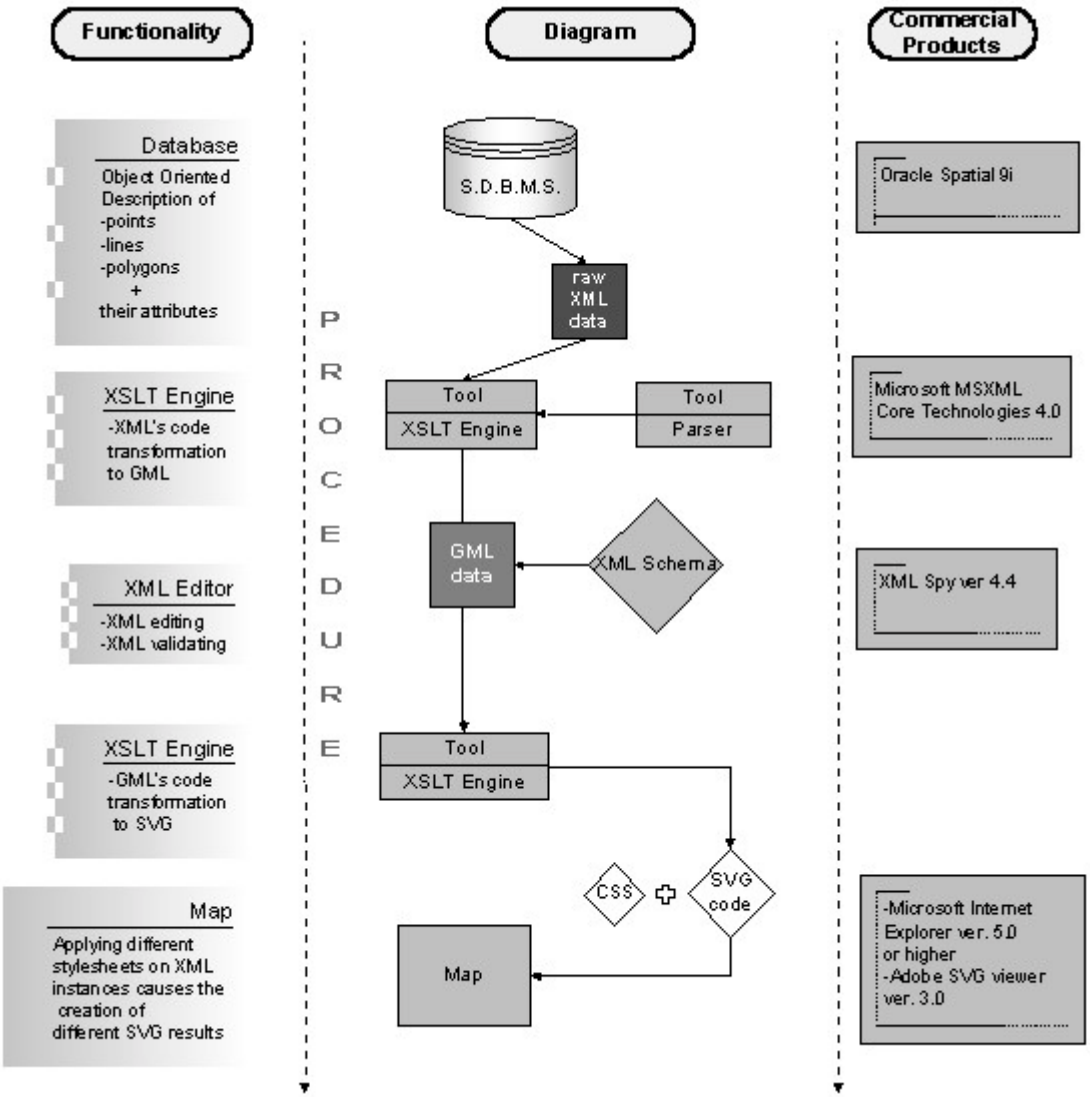

Figure 1. Flowchart for map composition through GML

A Unified Modelling Language (UML) class diagram is used to represent the conceptual data model (Figure 2). UML describes the design of class hierarchies and their interacting relationships. It is also able to include and express semantics of spatial data. The root element, as in any object-oriented database, is the *Object*. In particular, a spatial data database is made of Features. A *Feature* is an Object that has a spatial property of geometry type. Each Object has as a property a unique *Object ID*, which is inherited to the Feature. The geometry type according to the adopted OGC simple features standard can be: point, line and area. In response to the above, three specific feature types are created: *Point_Feature, Linear_Feature* and *Areal_Feature*. The Linear_Feature has the property *Length* and the *Areal_Feature*

the properties *Area* and *Perimeter*. According to this model each spatial data layer is transformed into a new Feature Type e.g. *Roads, Settlements, Administrative Units* that inherits the attributes of the *Point_Feature,* the *Linear_Feature* or the *Areal_Feature* in relation to its geometry. In addition, the specific *properties* of each layer appear in each new Feature Type e.g. *Settlements* have the properties *Code, Name and Population.*

2.2 Database Implementation

Utilizing the above-described schema, the project's database is implemented in Oracle 9i. Oracle 9i is not a true object-oriented DBMS, but an object-relational one. In the current version of Oracle Spatial, the object-relational model has been extended to include an SDO_GEOMETRY type, which is used to store the geometry of the objects. The geometry of the features is stored in Oracle using what the OGC calls *simple features* and composite collections of these primitive types. The same standards are also used in GML and in the database conceptual model. This compatibility is enviable, since the geo-database is developed to work as a repository where from the GML data is generated. Another advantage is Oracle's ability to communicate with well-known GIS systems such as ESRI's ARC/INFO. This factor facilitates the population of the database and the spatial data editing.

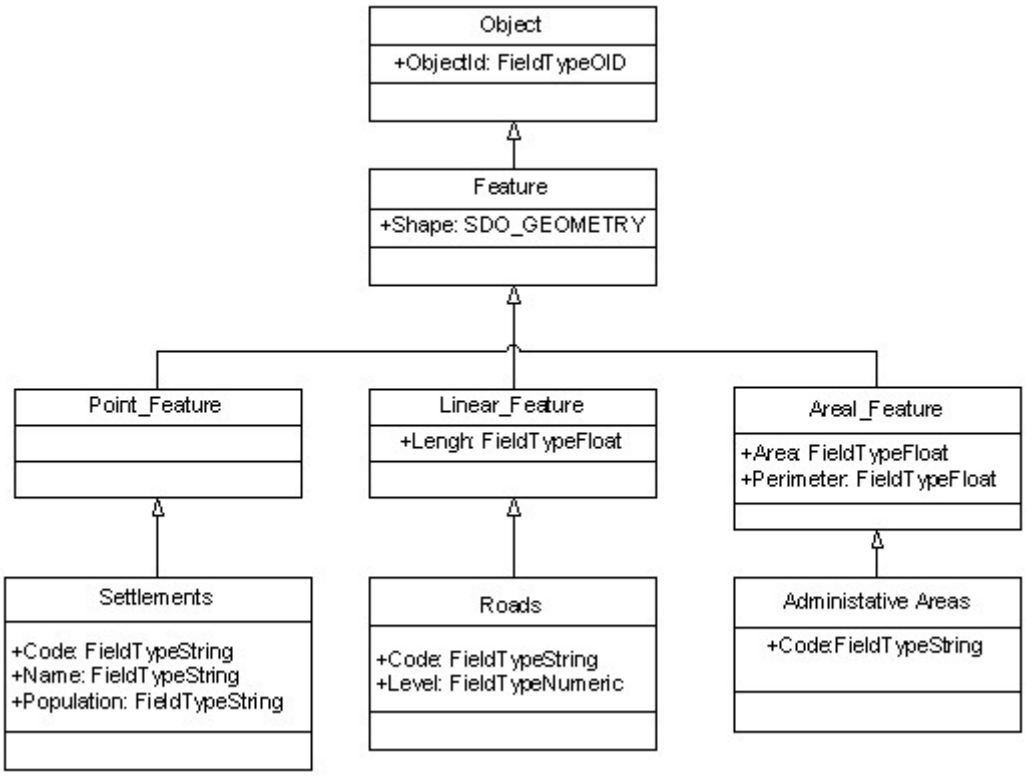

Figure 2. Part of the UML class diagram of the Spatial Data Model

Loading data into the spatial database is a straightforward process with the use of ARCSDE, which is an ARC/INFO client that works as a gateway between ARC/INFO and Oracle. For the creation of the spatial database, each layer stored as an ARC/INFO coverage, is converted to a Feature Type. The Feature Type preserves the geometry and all properties appearing in the feature attribute table of the coverage. A mapping exists between the geometry type of features in the geo-database and the coverages. In addition a new field called ObjectID is created, for all geometry types, which guarantees a unique ID for each object as required by the data model. All INFO tables that store additional properties of the features are transformed into Oracle tables. Coverage and INFO table items are mapped based on a combination of their type and their width.

3. GML DATA ENCODING

The Geography Markup Language (GML) is an XML data encoding scheme for the transport and storage of geographic information, including both the spatial and non-spatial properties of geographic features. GML has been designed to uphold the principle of separating content from presentation. It provides mechanisms for the encoding of geographic feature data regardless of the way the data may be presented to a human reader. GML is compliant with the XML Schema Recommendation and it has also been developed to be consistent with the XML Namespaces Recommendation. Namespaces are used to distinguish the definitions of features and properties defined in application-specific domains from one another, and from the core constructs defined in GML modules.

3.1 GML Application Schema

GML 2.0 defines three base schemas for encoding spatial information. Feature schema defines the general feature-property model, geometry schema includes the detailed geometry components, and xlinks schema provides the XLink attributes used to implement linking functionality. The GML Geometry schema includes type definitions for abstract geometry elements, concrete (multi) point, line and polygon geometry elements, as well as complex type definitions for the underlying geometry types. The Geometry schema targets the 'gml' namespace. A namespace is a conceptual entity identified by a URL that denotes a collection of element names and type definitions that comprise a cohesive vocabulary.

The three schema documents alone provide base types and structures, which may be used by an application schema. An application schema declares the actual feature and property types of interest for a particular domain, using components from GML in standard ways. Broadly speaking, these involve the definition of application-specific types, which are derived from types in the standard GML schemas, or direct inclusion of elements and types from the standard GML schemas. The base GML schemas effectively provide a meta-schema or a set of foundation classes, where from an application schema can be implemented. The UML for the GML application schema is depicted in Figure 3.

The <import> element in the Geometry schema brings in the definitions and declarations contained in the XLinks schema, while the <include> element in the Feature schema brings in the GML geometry constructs and make them available for use in the definition of feature types.

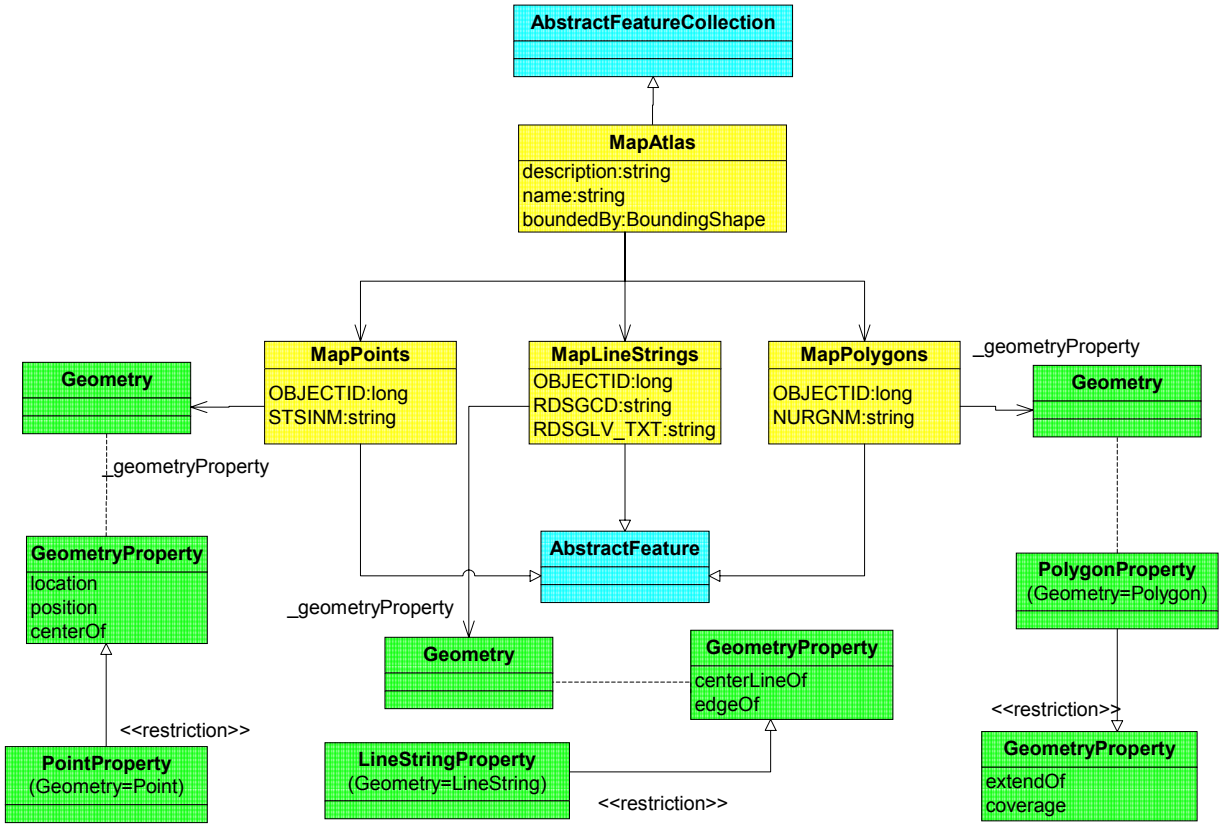

Figure 3. UML for the GML Application Schema

3.2 GML generation and XSLT

The XML data for each fundamental type of spatial geometry (points, lines and polygons) is extracted from the spatial data database through spatial queries (Table 1). The XML data is converted to the GML form, in accordance with the project specific application *schema* utilizing XSLT. Since GML is an XML application, it can be readily styled into a variety of presentation formats including vector and raster graphics, text, sound and voice. Generation of graphical output such as maps is one of the most common presentations of GML and this can be accomplished using the *eXtensible Stylesheet Language Transformation* (XSLT).

Table 1. Sample XML file for points geometry as extracted from Oracle 9i

```
<layer id="cities">
    <ROW num="1">
        <OBJECTID>5095</OBJECTID>
        <STSINM>RETHIMNON</STSINM>
        <SHAPE>
            <SDO_GTYPE>2001</SDO_GTYPE>
            <SDO_POINT>
                <X>1412195.02330894</X>
                <Y>-1812727.78015544</Y>
            </SDO_POINT>
        </SHAPE>
    </ROW>
    <ROW num="2">
        <OBJECTID>5009</OBJECTID>
        <STSINM>AYIOS NIKOLAOS</STSINM>
        <SHAPE>
            <SDO_GTYPE>2001</SDO_GTYPE>
            <SDO_POINT>
                <X>1527197.010353</X>
                <Y>-1807571.19085402</Y>
            </SDO_POINT>
        </SHAPE>
          </ROW>
          :
</layer>
```

XSLT provides a set of tools for transforming documents from their original structure to a new structure and is commonly described as a transformative style language. Instead of adding information to the original document structure, it creates a new document structure based on rules *(stylesheets)* applied to the content of the original. An XSLT stylesheet is a collection of template rules (Table 2 and Table 6). Each template rule includes a pattern that identifies the source tree nodes to which the pattern applies and a template that is added to the resulting tree when the XSLT processor applies a specific template rule to a matched node (4). For example, the same XML document can be transformed to GML, SVG or HTML by applying to the source document three different stylesheets (Table 3).

Table 2. Sample Stylesheet for transforming XML to GML data (points geometry)

```
<xsl:stylesheet version="1.0" ..... >
<xsl:template match="/">
<MapAtlasLab="Cartography"Date="2002-07-25"
xsi:noNamespaceSchemaLocation="E:\Spanaki\GML\XML2GML\MapAtlas_4POINTS.xsd">
    <xsl:for-each select="layer/ROW">
        <MapPoints>
            <xsl:attribute name="STSINM"><xsl:value-of select="STSINM"/></xsl:attribute>
            <xsl:attribute name="OBJECTID"><xsl:value-of select="OBJECTID"/></xsl:attribute>
            <PointsGeometry>
                <gml:coord>
                    <gml:X>
                        <xsl:apply-templates select="SHAPE/SDO_POINT/X"/>
                    </gml:X>
                    <gml:Y>
                        <xsl:apply-templates select="SHAPE/SDO_POINT/Y"/>
                    </gml:Y>
                </gml:coord>
            </PointsGeometry>
        </MapPoints>
        </xsl:for-each>
    </MapAtlas>
</xsl:template>
</xsl:stylesheet>
```

Table 3. Output GML sample file for points geometry

```
<?xml version="1.0" encoding="UTF-8"?>
<MapAtlasLab="Cartography"Date="2002-07-25"
xsi:noNamespaceSchemaLocation="E:\Spanaki\GML\XML2GML\MapAtlas_4POINTS.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <MapPoints STSINM="RETHIMNON" OBJECTID="5095">
        <PointsGeometry>
            <gml:coord>
                <gml:X>1412195.02330894</gml:X>
                <gml:Y>-1812727.78015544</gml:Y>
            </gml:coord>
        </PointsGeometry>
    </MapPoints>
    <MapPoints STSINM="AYIOS NIKOLAOS" OBJECTID="5009">
        <PointsGeometry>
            <gml:coord>
                <gml:X>1527197.010353</gml:X>
                <gml:Y>-1807571.19085402</gml:Y>
            </gml:coord>
        </PointsGeometry>
          </MapPoints>
          :
</MapAtlas>
```

XSLT can be considered as a programming language for the process and transformation of XML documents. As such, it supports (5):

- A small set of flexible data types: Boolean, number, string, node-set, and external objects
- A full set of operations: <xsl:template>, <xsl:apply-templates>, <xsl:sort>, <xsl:output>
- Programming flow-control: <xsl:if>, <xsl:for-each>, <xsl:choose>

XSLT is affiliated with *XML Path Language* (XPath), which is a specialized language for addressing parts of an XML document. XPath provides a simple and concise syntax for the identification of nodes in an XML document, based on the node's type, name, content and context in relation to other nodes in the tree. XSL Transformations (XSLT) provides a grammar in which the results of XPath queries are associated with templates to describe the transformation - parsing of data in the XML source document as a new XML document. Like the core of the SQL languages used to manipulate relational databases, XPath is important because it gives a flexible way to identify the desirable information that someone might want to pull out from a larger collection.

3.3 XSLT processing

XSL transformations are implemented using XSLT processors. An XSLT processor is called with a command prompt (e.g. the DOS command prompt in a Windows operating system) and requires two basic pieces of information: the source document to be read into the source tree and the stylesheet to apply to the source tree document (Figure 4).

More specifically, XSLT processing involves the following steps (6):

- Reading the XML source document and the associated XSLT stylesheets
- Parsing XML files, and then their associated XSLT files, into XML Document Object Model (DOM) objects that represent the XML elements as trees of nodes. Each node corresponds to an XML document element or attribute
- Applying XSLT transformation to the source trees
- Producing resulting trees in accordance with the specification of the XSLT style sheet
- Serializing the resulting trees as output files. These files output XML, GML, SVG, or other formats
- The XSL parser, used in the framework of this application, is the Microsoft's MSXSL parser which is a very efficient open source product. The XSLT is employed again for the transformation of GML data into SVG graphic output.

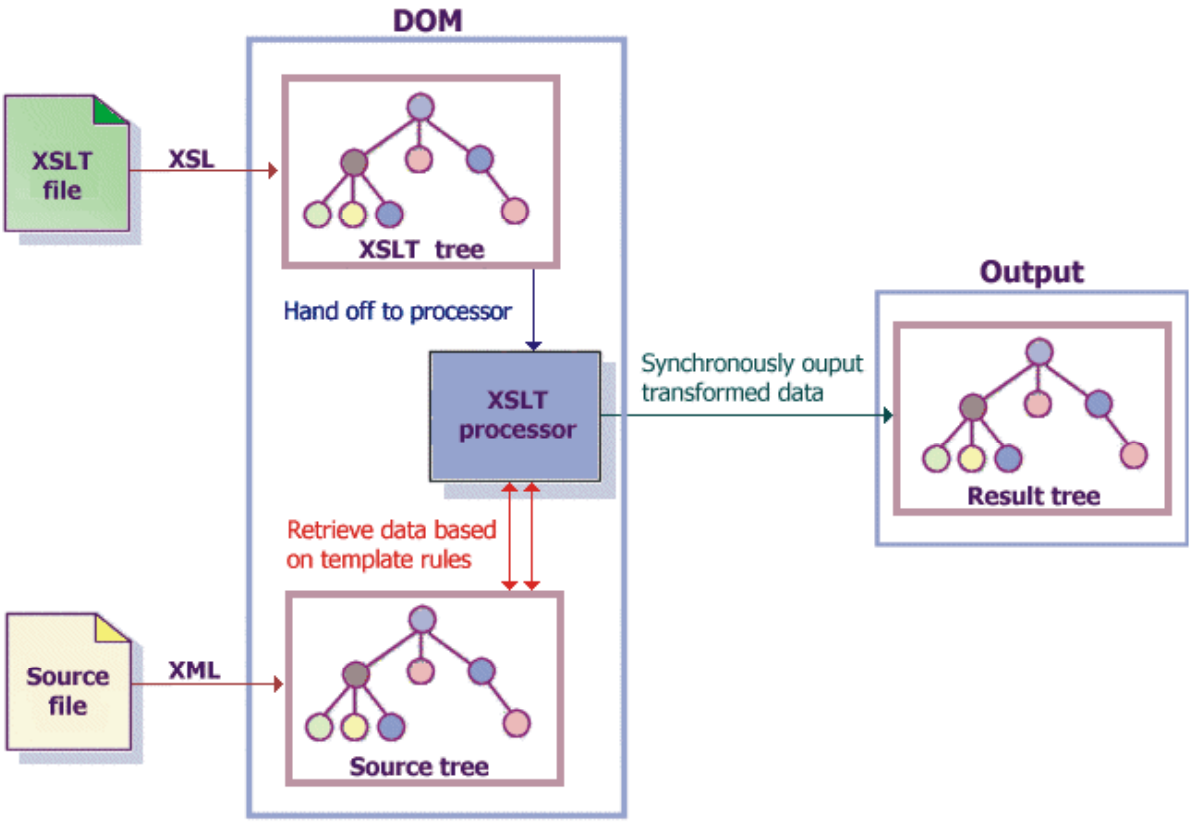

Figure 4. XSLT processing of XML-formatted documents

4. MAP COMPOSITION

GML data display is accomplished through the SVG's (Scalable Vector Graphics) graphic format. SVG is a two-dimensional vector graphics format written in XML. As an XML format, SVG is platform independent and non-proprietary.
SVG has been developed for two basic reasons (7):

- The perception that with the arrival of parts of the XML family of technologies, the need and an exciting opportunity existed for an XML-based graphics format
- The long-standing recognition of the limitations of traditional bitmap graphics.

In particular, one of the limiting factors of using images on the Web has been that they are limited to the display of 256 (or 216) colours, usually of the 'web-safe' palette. In contrast, SVG can use 16.7 million colours (24 bit). SVG reaches the computer as text-based source code, which is then rendered on the client side. An SVG rendering engine, also known as an SVG viewer, interprets the source code and interpolates the true colour needed in a gradient or other colour transition, thus providing high-quality, true-to-life (or true-to-code) colour.

The most important advantages of SVG are (7, 8)

- SVG is searchable. Text and elements in it may be indexed by the search engine software. Using a search engine, users may look for a map that has a particular street name on it. Then they can search that map for the location of the street name on the map. To do the equivalent with a raster format, someone would need optical character recognition software
- SVG graphics are resolution and device independent (can be scaled to match different devices)
- SVG format creates smaller size files that reduce download times compared to bitmapped graphics. That's why they are better suited for devices with low bandwidth and limited memory
- SVG images can be panned and zoomed without degrading image quality
- SVG specification is fully compatible with existing technologies like HTML and XHTML, XLink, XML Namespaces, DOM, CSS and XSL
- SVG graphics can use scripting to provide interactivity and animation

- SVG images can be animated by two methods: the use of declarative SVG elements (mostly using elements borrowed from SMIL Animation) or the use of ECMAScript (JavaScript) or another scripting language to manipulate the Document Object Model (DOM) of an SVG image.

Table 4. Basic SVG elements

Element	Basic Attributes	Notes / Description
Rect	x, y, width, height	Rectangles and rounded rectangles
Circle	cx, cy, r	Center (x,y) and radius
ellipse	cx, cy, rx, ry	Center (x,y) and x/y-axis radius of the ellipse
line	x1, y1, x2, y2	Start and endpoints
polyline	Points list	"x-y-coordinate-pairs" The points that make up the polyline. Space- or comma-separated. Example: "20,20 50,100 200,80 70,300"
polygon	Points list A polygon is exactly the same as a polyline, except that the figure is automatically closed.	"x-y-coordinate-pairs" The points that make up the polygon. Space- or comma-separated
path	d attribute	A path is a more complex vector definition that can contain straight segments and curves The d attribute contains the path definition (segments and curves)
text	String, x, y	
image	x, y, width, height,xlink:href	This allows embedding PNG or JPEG (but *not* a GIF) images within an SVG image.

Perhaps the only disadvantage of SVG graphics is that they require the installation of a viewer in order to be displayed in a Web browser environment (e.g. Adobe SVG viewer). The need for a plug-in with SVG is temporary, as this recommendation is expected to be supported natively by user agents through the implementation of XML on the Internet.

4.1 Graphical rendering

Table 4 describes the SVG elements' basic attributes that should be considered when transforming GML files to SVG. The correspondence between the GML geometry types and the SVG structures is easily implemented as they refer to the same geometric constructs (9). Note that a Point will be represented with a rectangle, a circle or a more complex pattern more figurative of the object. The Table 5 with the transformations follows:

Table 5. Geometry object correspondence between GML and SVG

GML	SVG
<point>	rectangle, circle, icon at the point location
<LineString>	<polyline>
<Box>	<rect>
<LinearRing>	<svg:polygon>
<Polygon>	<polygon>
<MultiPoint>	rectangle, circle, icon at the point
<MultiLineString>	A group <g> of <svg:polyline>
<MultiPolygon>	A group <g> of <svg:polygon>

Table 6. Sample Stylesheet for transforming GML data to SVG graphics

```
<xsl:stylesheet version="1.0" ..... >
<xsl:template match="/">
    <xsl:for-each select="MapAtlas/MapPoints">
        <xsl:variable name="cx" select="format-number(PointsGeometry/gml:coord/gml:X,'#')"/>
        <xsl:variable name="cy" select="format-number(PointsGeometry/gml:coord/gml:Y,'#')"/>
        <xsl:variable name="circle_X" select="($cx)"/>
        <xsl:variable name="circle_Y" select="($cy)"/>
        <g transform="scale(1,-1)">
            <xsl:attribute name="id">
            <xsl:value-of select="@OBJECTID" />
              </xsl:attribute>
            <xsl:attribute name="Label">
            <xsl:value-of select="@STSINM" />
             </xsl:attribute>
             <circle style="fill:red; stroke:black; stroke-width:100" r="2500" cx="{$circle_X}" cy="{$circle_Y}"/>
            <text  dx="3000" x="0" y="0" style="font-family:Arial; stroke:black; stroke-width:0.001; font-size:6000;
fill:rgb(230,194,0); " >
                <xsl:attribute name="transform">translate(<xsl:value-of select="format-
number(PointsGeometry/gml:coord/gml:X,'#')"/>,<xsl:value-of select="format-
number(PointsGeometry/gml:coord/gml:Y,'#')"/>)scale(1,-1)</xsl:attribute>
                <xsl:value-of select="@STSINM" />
            </text>
        </g>
    </xsl:for-each>
</svg>
</xsl:template>
</xsl:stylesheet>
```

XSLT is used again to perform the transformations that lead to the visualization of GML data through SVG graphics. The generated SVG document has a structure similar to the one of the GML document. The ‘g’ element in an SVG document, groups and names collections of drawing elements. If several drawing elements share similar attributes, they can be collected together using a ‘g’ element. A group of drawing elements (‘g’), as well as individual objects can be given a name using the ‘id’ attribute (Table 7).

4.2 SVG and CSS

For the creation of SVG graphic output through the transformation of XML files, the *Cascading Style Sheets – CSS* can be used. CSS is commonly described as an annotative style language, meaning that it adds formatting information to the document tree rather than changing the document itself (10). CSS don’t use XML or XSL syntax, formatting with a different way the final graphic result. CSS is easier to use than XSLT and uses less memory because it cannot reorder a document or build a tree representation of the document.

Table 7. Sample of SVG document for Points geometry

```
<?xml version="1.0" encoding="UTF-16"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/2001/REC-SVG-
20010904/DTD/svg10.dtd">
<svg width="100%" height="100%" viewBox="1290000 1750000 300000 120000"
xmlns:gml="http://www.opengis.net/gml">
    <g transform="scale(1,-1)" id="5095" Label="RETHIMNON">
        <circle style="fill:red; stroke:black; stroke-width:100" r="2500" cx="1412195" cy="-1812728"/>
        <text dx="3000" x="0" y="0" style="font-family:Arial; stroke:black; stroke-width:0.001; font-size:6000;
fill:rgb(230,194,0); " transform="translate(1412195,-1812728)scale(1,-1)">RETHIMNON</text>
    </g>
    <g transform="scale(1,-1)" id="5009" Label="AYIOS NIKOLAOS">
        <circle style="fill:red; stroke:black; stroke-width:100" r="2500" cx="1527197" cy="-1807571"/>
        <text dx="3000" x="0" y="0" style="font-family:Arial; stroke:black; stroke-width:0.001; font-size:6000;
fill:rgb(230,194,0); " transform="translate(1527197,-1807571)scale(1,-1)">AYIOS NIKOLAOS</text>
    </g>
    :
</svg>
```

The use of SVG and the appropriate stylesheet can be utilized for rendering topographic and thematic maps like the one shown in Figure 5, which depicts the Population Density in the European Union. Special stylesheets can be designed to implement a variety of cartographic symbol sets thus providing libraries of symbols or alternative symbolization approaches. Since the content is separated from presentation, different stylesheets can produce alternative symbolization scenarios for the same data set according to the users' preferences.

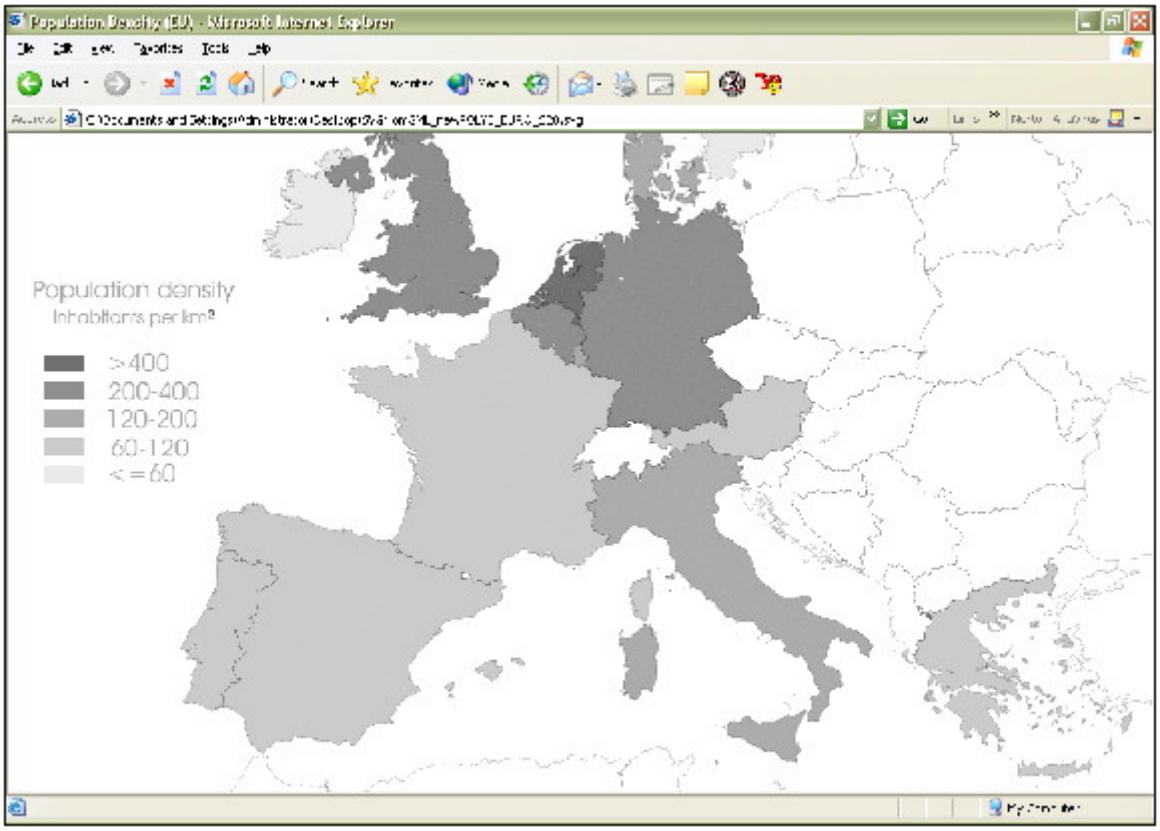

Figure 5. Thematic map on population density composed with a stylesheet

SVG supports the following methods for the implementation of CSS styling (11):

- External CSS stylesheets referenced from the current document using a processing instruction from XSLT code
  e.g. <?xml-stylesheet type="text/css" href="region.css" ?>
- Internal CSS stylesheets (i.e., style sheets embedded within the current document, such as within an SVG 'style' element)
  e.g. <style type="text/css">
  #LASITHI{ fill:rgb(203,194,153); opacity:0.2 }
  #IRAKLEIO{ fill:rgb(137,205,190); opacity:0.5 }
  </style>
- Inline style (i.e., CSS property declarations within a style attribute on every particular SVG element generated as output from XSLT transformations)
  e.g. <circle style="fill:red; stroke:black; stroke-width:100" r="2500" cx="{$circle_X}" cy="{$circle_Y}"/>

The concept of linking the XSL stylesheet of CSS files externally has been adopted as the optimal solution, on the grounds of ensuring the independence of the data transformation procedure from the drawing process.

5. CONCLUSIONS

XML family of technologies constitutes a revolution in the way geographic information is encoded, transferred and rendered. XML is not a language in a normal sense. It is rather a meta-language designed to create a variety of what one would normally think of as languages, of which SVG is one example. Because these different languages are all built in the same syntax, powerful new possibilities exist for interchanging information between computer applications, due to the fact that the structure of XML documents is predictable. SVG provides the power and flexibility to create Web based and paper based graphics, whose inherent characteristics supercede the limitations of traditional bitmap graphics. It was designed to be open to creation or implementation by programmers or by those who can think like programmers using graphics drawing tools that are likely to be familiar. In addition, SVG is accessible to a wide range of devices

from small mobile devices through office computer monitors to high resolution printers. These capabilities open new ways in the design of maps and charts and the portrayal of geographic features.

6. REFERENCES

[1] OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.1, OpenGIS® Implementation Specification, OpenGIS Project Document Number 02-009, http://www.opengis.net/gml/02-009/GML2-11.html, 25 April 2002

[2] R. Walker, ed. AGI Standards Committee GIS Dictionary, Association for Geographic Information) (1993)

[3] OpenGIS® Abstract Specification (http://www.opengis.org/techno/specs.htm) (1999)

[4] DuCharme, B., "XSLT Quickly", Manning Editions, (2002)

[5] World Wide Web Consortium, "XSL Transformations (XSLT)" Version 1.0, (1999)

[6] Microsoft Corporation, Microsoft XML Parser User's Manual, Microsoft XML Core Services (MSXML) 4.0, (2000)

[7] Watt A., Designing SVG web graphics, New Riders Publishing, (2002)

[8] DBx Geomatics, "SVG Map Maker for MapInfo Professional Software Documentation", Scalable Vector Graphics (SVG) Publish vector maps on the web, (2002)

[9] Taladoire G., Geospatial data integration and visualisation using open standard", (2001)

[10] World Wide Web Consortium, "Cascading Style Sheets level 2 CSS2 Specification", (1998)

[11] World Wide Web Consortium, "Associating Style Sheets with XML documents" Version 1.0., (1999)

AN XML-BASED APPROACH FOR THE COMPOSITION OF MAPS AND CHARTS

Tsoulos, L., Spanaki, M. and Skopeliti, A.

Cartography Laboratory, School of Rural and Surveying Engineering, National Technical University of Athens
9 H. Polytechniou, 157 80 Zographou Campus, Athens, Greece.
Tel: +30 -210-7722730. Fax: +30-210-7722734. E-mail: lysandro@central.ntua.gr, spanaki@mail.ntua.gr
and askop@survey.ntua.gr

Biography

Prof. Lysandros TSOULOS graduated from the Department of Rural and Surveying Engineering (University of Thessalonica) in 1972. He received his Dr. Eng. from the National Technical University of Athens in 1989. His dissertation entitled "The Electronic Chart and its Use in Navigation" contributed to the standardization of the content and functionality of electronic charts on board the ships. He worked for the Hellenic Navy Hydrographic Service for 16 years as director of the directorate of Cartography and the Computing Center.

Currently he is an Associate Professor at the Department of Rural and Surveying Engineering, Vice- Chairman of the Department and Director of the NTUA Geomatics Center. He has 60 publications in international journals and conference proceedings on Digital Mapping and Geographical Information Systems. He has also participated in a number of EU funded projects like: Impact II (Environmental and Social Impact of Air Quality), STRIDE, MEDSPA (Management of Ecosystems), WERATLAS (Atlas of wave energy resource in Europe), GEOMED (Geographical Mediation System), Eurostat/GISCO (Cartography and Map Design) and STATLAS (Statistical Atlas of the European Union). He is member in a number of National and International Scientific Committees.