

# **Geospatial data integration and visualisation using open standard**

**Gilles TALADOIRE**

**Université de La Nouvelle Calédonie**  
**BP 4477- 98847 NOUMEA CEDEX – New Caledonia**  
[gt@univ-nc.nc](mailto:gt@univ-nc.nc)

**Invited until june 2001 at :**  
**Laboratoire I3S - UNSA – CNRS**  
**Les Algorithmes – 2000, route des Lucioles - Sophia Antipolis - 06410 BIOT**

## **Abstract :**

Lot of systems participate in the management of environmental data. This systems can be broken down into :

- acquisition from aerial images, satellite images, ground sensor, ...
- data archiving including conception, storage and extraction,
- data processing and analysis,
- data visualisation for each kind of user, graphically on the web.

Very often these systems are independent and end in incompatibility or incoherence in this processing chain. Our project is to propose an architecture based on the “Geographic Markup Language” (GML) to transmit the data from a system to another from acquisition to visualisation.

After a presentation of the global project, we describe the data description languages GML and SVG. We present the correspondence between the GML and SVG data structures and the styling information to add for map display. Indeed, we need to transform features and properties of GML into graphic objects of SVG with different aspect. Then, we show the different solutions to implement this mapping : XSL (“eXtensible Style Language”), Java using DOM (“Document Object Model”). To conclude, we present the many extensions to study to go further in this way : the addition of interactivity and animation, the display of the links between objects, the conception of a style sheet editor and generator using the XML schema of the instance document.

## 1 Introduction

First, we will present the context of our work with the localisation and the participants. Then, after a presentation of the global project, we describe the data description languages GML and SVG. We present the correspondence between the GML and SVG data structures and the styling information to add for map display. Indeed, we need to transform features and properties of GML into graphic objects of SVG with different aspect. Then, we show the different solutions to implement this mapping : XSL (“eXtensible Style Language”), Java using DOM (“Document Object Model”). To conclude, we present the many extensions to study to go further in this way.

## 2 Context

The works are managed by the "Université de La Nouvelle Calédonie" (University of New Caledonia - UNC) and the "Institut de Recherche pour le Développement" (IRD).



The UNC is a very small French university in New Caledonia, an island in south pacific between Australia and New Zealand. The computer sciences team work on the theme "Computer Sciences for the environment". For that, they collaborate with geographer and geologist to integrate multidisciplinary knowledge for the management of the environment.



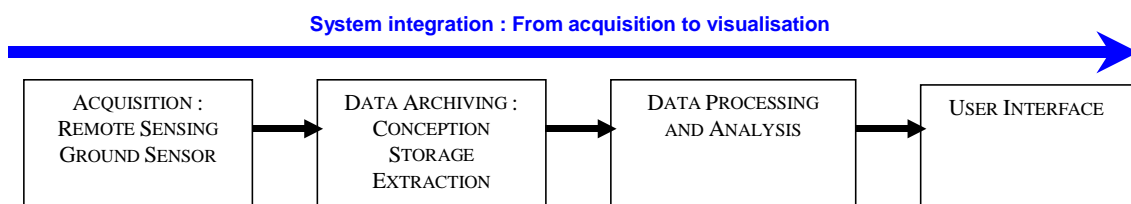
The IRD is a French multidisciplinary research institute with 3 main themes in New Caledonia : marine environment, terrestrial environment and society. His research focuses on natural and human environments in tropical island. The different locations of the research unit ESPACE constitute a very powerful network for tropical study and satellite reception station : French Guyana (South America), La Réunion (Indian Ocean), Orléans (France), Montpellier (France) and New Caledonia (Pacific Ocean).

The goal of the collaboration between UNC and IRD is to develop Environmental Information System for a sustainable development.

## 3 Environmental Information System

Lot of systems participate in the management of environmental data. This systems can be broken down into :

- acquisition from aerial images, satellite images, ground sensor, ...
- data archiving including conception, storage and extraction,
- data processing and analysis,
- data visualisation for each kind of user, graphically on the web.



**Figure 3.1 : System integration : From acquisition to visualisation**

Very often these systems are independent and end in incompatibility or incoherence in this processing chain. Automation is then difficult or impossible. More over, generally many

partners are involved in environmental projects (expert firm, administration, research institute, ...). Probably, they don't use the same methods, the same software in a single place. We have heterogeneous data on many places. With this observation, we need to propose methods to connect together this different information systems along the process : from acquisition to visualisation.

Let us described briefly each component of the global process.

**Data acquisition** is the beginning of any environmental process. This acquisition can be done with satellite sensor, with aerial sensor, with ground sensor. Each application or material give data without standard.

**Data archiving** is fundamental for the data preservation and usage. This process includes the conceptual modelling, the storage and the extraction of the data. Different models are used, different system are used (Geographical Information System - GIS or Data Base Management System - DBMS) with their particularities.

**Data processing and analysis** is the useful part. The data archiving must be used to supply data to other systems and to be supplied with process data. The systems to be connected can be expert systems, neural networks, multi-agents systems, data mining systems, image processing and analysis, simulations, scientific model, ... Each system has his own data format.

**Visualisation** is the final process. The visualisation must be adapted to the different users of an environmental project : scientific and expert, decision maker, teacher and student, general public, ... The information must be available on Internet, usually with graphics.

We note that we need to make collaborate these different systems to use the data efficiently and to automate operations like acquisition, archiving, processing, ...

#### **4 Existing initiatives**

**In the domain of the internet**, we note the World Wide Web Consortium (W3C) initiative. The W3C develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential as a forum for information, commerce, communication, and collective understanding. Among this technologies, we are interested by the Extensible Markup Language (XML) [XML10] which is the universal format for structured documents and data on the Web. An application of XML developed by the W3C for describing two-dimensional graphics is SVG (Scalable Vector Graphics) [SVG].

**In the domain of geospatial information**, we note two standardisation initiatives which collaborate together for some points : the OpenGIS consortium [OGC] and the technical comity ISO 211 [ISO211].

The aims of the Technical Comity ISO 211 is the standardisation in the field of digital geographic information. For that, it will be established a structured set of standards for information concerning objects or phenomena that are directly or indirectly associated with a location relative to the Earth.

The objective of the OpenGIS project is to define transparent access to heterogeneous geodata and geoprocessing resources in a networked environment. For that, the OpenGIS Project will provide a comprehensive suite of open interface specifications that enable developers to write interoperating components that provide these capabilities. Among these specifications, we are interested by GML (Geographic Markup Language) [GML20].

For later use in the document, we describe shortly GML and SVG.

#### 4.1 GML (Geographic Markup Language)

The Geography Markup Language (GML) [GML20] (or in French [GML20f]) is an XML encoding for the transport and storage of geographic information, including both the spatial and non-spatial properties of geographic features. This specification defines the XML Schema syntax, mechanisms, and conventions that :

- Provide an open, vendor-neutral framework for the definition of geospatial application schemas and objects;
- Allow profiles that support proper subsets of GML framework descriptive capabilities;
- Support the description of geospatial application schemas for specialized domains and information communities;
- Enable the creation and maintenance of linked geographic application schemas and datasets;
- Support the storage and transport of application schemas and data sets;
- Increase the ability of organisations to share geographic application schemas and the information they describe.

Implementers may decide to store geographic application schemas and information in GML, or they may decide to convert from some other storage format on demand and use GML only for schema and data transport.

The OGC Abstract Specification [OGC99] defines a geographic feature as "*an abstraction of a real world phenomenon; it is a geographic feature if it is associated with a location relative to the Earth*". Thus a digital representation of the real world can be thought of as a set of features. The state of a feature is defined by a set of properties, where each property can be thought of as a {name, type, value} triple. The number of properties a feature may have, together with their names and types, are determined by its type definition. Geographic features are those with properties that may be geometry-valued. A feature collection is a collection of features that can itself be regarded as a feature; as a consequence a feature collection has a feature type and thus may have distinct properties of its own, in addition to the features it contains.

GML 2.0 defines three base schemas for encoding spatial information :

- The Geometry schema (geometry.xsd) for the description of the Geometry represented with UML in Figure 4.1,
- The Feature schema (features.xsd) for the description of the Features represented in Figure 4.2,
- The Xlink schema (xlinks.xsd) until the Xlink schema will be published by the W3C.

A conformant GML application schema must import the Feature schema as shown in Figure 4.3.

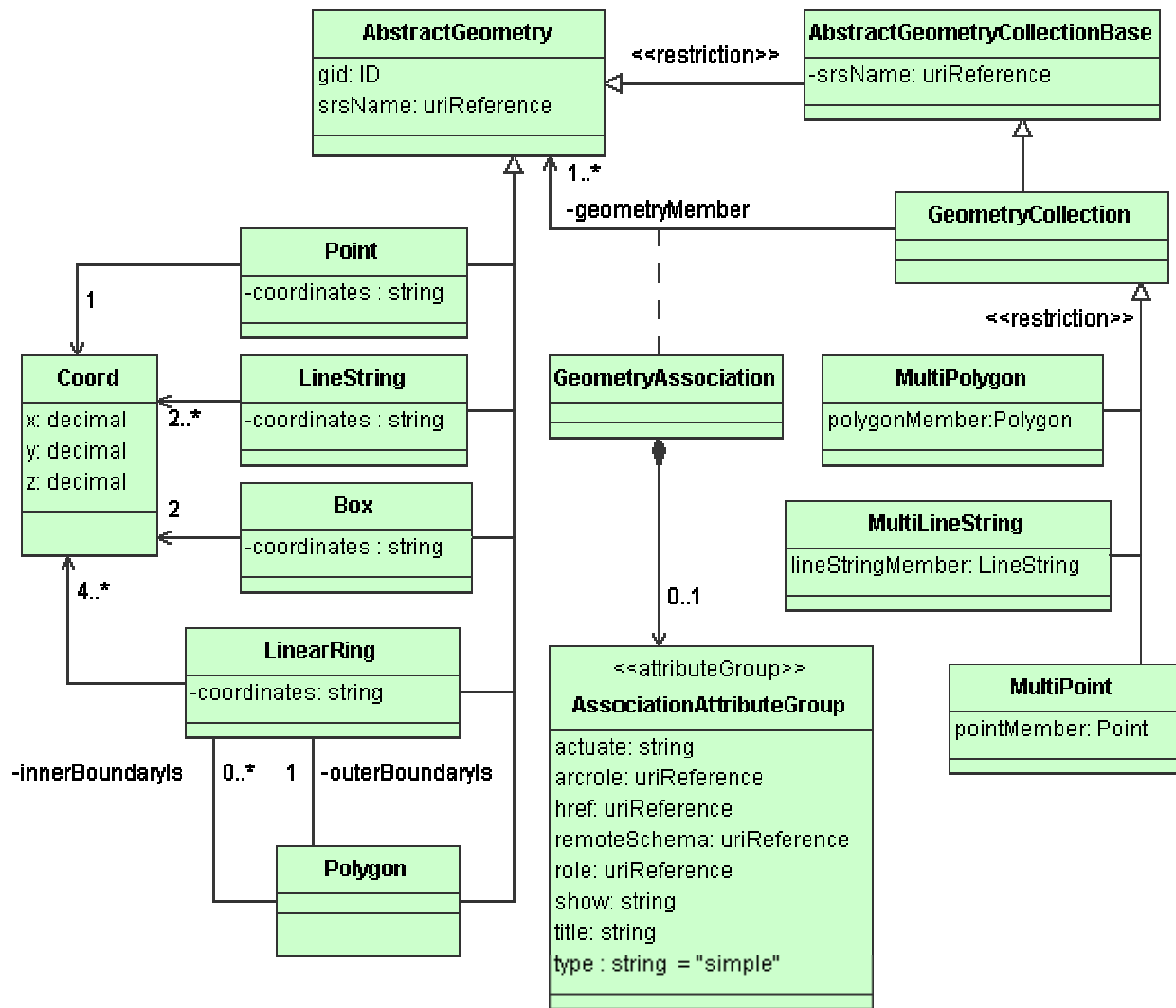


Figure 4.1 : UML representation of the Geometry schema



## 4.2 SVG (Scalable Vector Graphics)

SVG [SVG] is a language for describing two-dimensional graphics in XML [XML10]. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects and template objects. SVG drawings can be interactive and dynamic. Animations can be defined and triggered either declaratively (i.e., by embedding SVG animation elements in SVG content) or via scripting. Sophisticated applications of SVG are possible by use of supplemental scripting language with access to SVG's Document Object Model (DOM), which provides complete access to all elements, attributes and properties. A rich set of event handlers such as onmouseover and onclick can be assigned to any SVG graphical object.

SVG stands for Scalable Vector Graphics :

**Scalable** : To be scalable means to increase or decrease uniformly. In terms of graphics, scalable means not being limited to a single, fixed, pixel size. On the Web, scalable means that a particular technology can grow to a large number of files, a large number of users, a wide variety of applications. SVG, being a graphics technology for the Web, is scalable in both senses of the word.

**Vector** : Vector graphics contain geometric objects such as lines and curves. This gives greater flexibility compared to raster-only formats (such as PNG and JPEG) which have to store information for every pixel of the graphic. Typically, vector formats can also integrate raster images and can combine them with vector information such as clipping paths to produce a complete illustration; SVG is no exception.

**Graphics** : Most existing XML grammars represent either textual information, or represent raw data such as financial information. They typically provide only rudimentary graphical capabilities, often less capable than the HTML 'img' element. SVG fills a gap in the market by providing a rich, structured description of vector and mixed vector/raster graphics; it can be used standalone, or as an XML namespace with other grammars.

The structure of a SVG document is described by this part of the DTD :

```
<!ELEMENT svg (desc|title|metadata|defs|
path|text|rect|circle|ellipse|line|polyline|polygon|
use|image|svg|g|view|switch|a|altGlyphDef|
script|style|symbol|marker|clipPath|mask|
linearGradient|radialGradient|pattern|filter|cursor|font|
animate|set|animateMotion|animateColor|animateTransform|
color-profile|font-face
%ceExt;%svgExt;)* >
```

We can simplify to keep only what we use forward and we consider that the structure of a SVG document looks like shown in Figure 4.4.

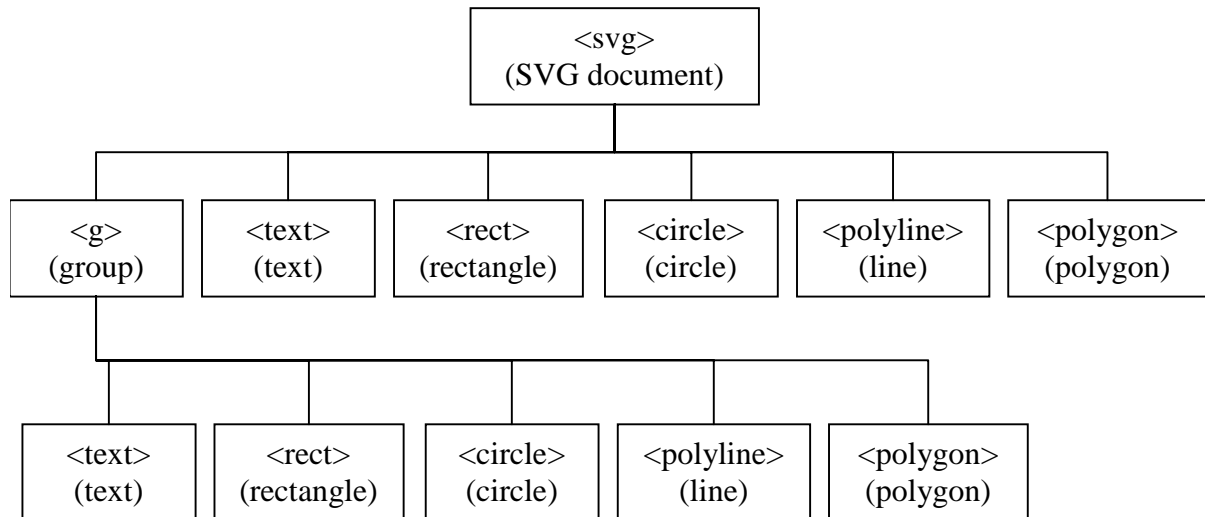


Figure 4.4 : Very simplified structure of a SVG document

## 5 Architecture for system integration

Our Project is to propose an architecture based on the “Geographic Markup Language” (GML) to transmit the data from a system to another.

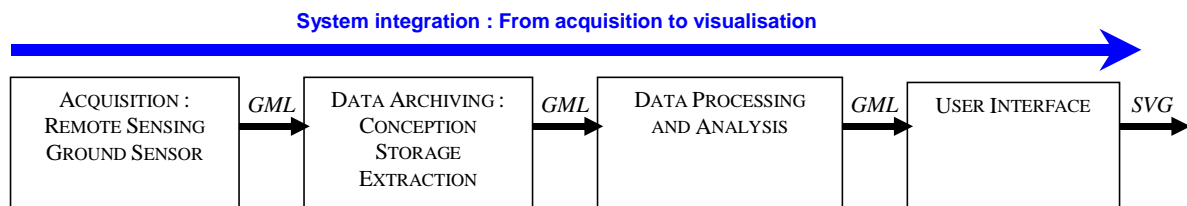


Figure 5.1 : How to realise the data integration

For that, we propose to construct for each system import/export modules which use the GML application schema. The import module will be able to transform GML data into system object (Figure 5.2). The export module that will be able to transform system object into GML data (Figure 5.3).

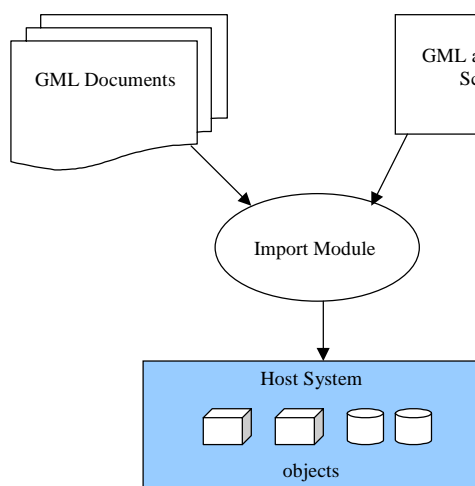


Figure 5.2 : GML data Import

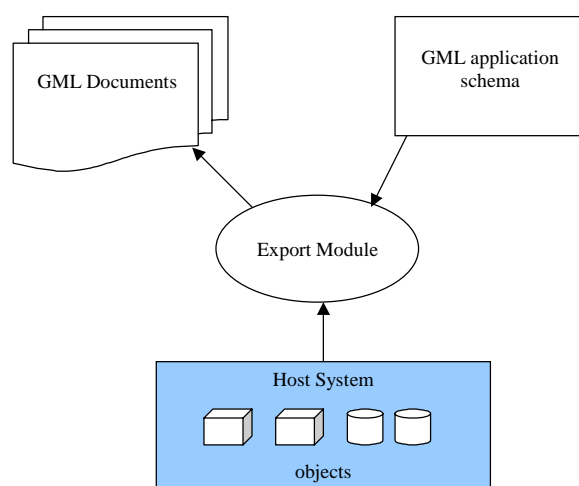


Figure 5.3 : GML data Export

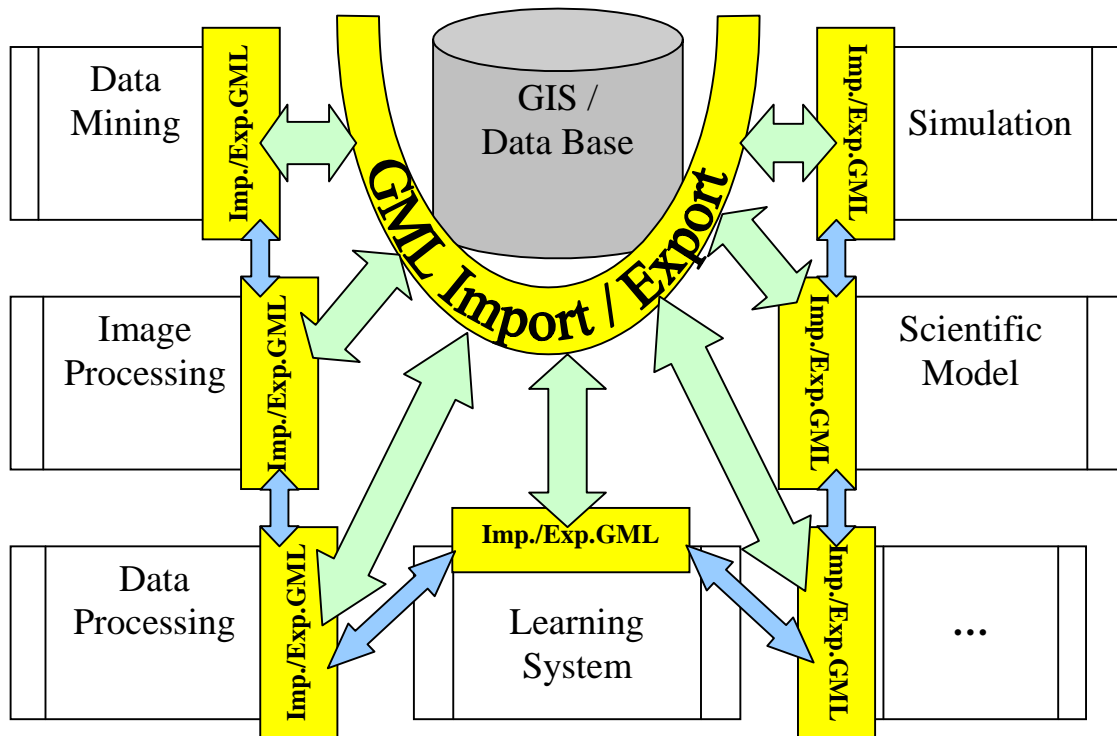


Figure 5.4 : System Interface with GML import / export modules

The OpenGIS consortium specify the functionality of the server side in the Web Feature Server specification (not yet approved). We have to develop the other import/export module in a generic way to be used by many applications et by many external systems.

This work has to be done. We suppose that all spatial data could be accessible as GML data. We will focus now on the visualisation process of this GML data.

## 6 Visualisation : from GML to SVG

GML has been designed to uphold the principle of separating content from presentation. GML provides mechanisms for the encoding of geographic feature data without regard to how the data may be presented to a human reader. Generation of graphical output such as maps can be accomplished in styling into an XML graphics technology (e.g. SVG [SVG] or X3D [VRML200x]).

We choose GML to represent spatial data and SVG to describe graphical data. Our work is to propose an automatic way to transform GML document in SVG (“Scalable Vector Graphic”) document to save user repetitive task. We have to determine if the information available are sufficient and, if needed, add what might be useful to generate the graphic. We are going to describe the transformations to do with the base structure, then determine the viewing style and generate the final graphic.

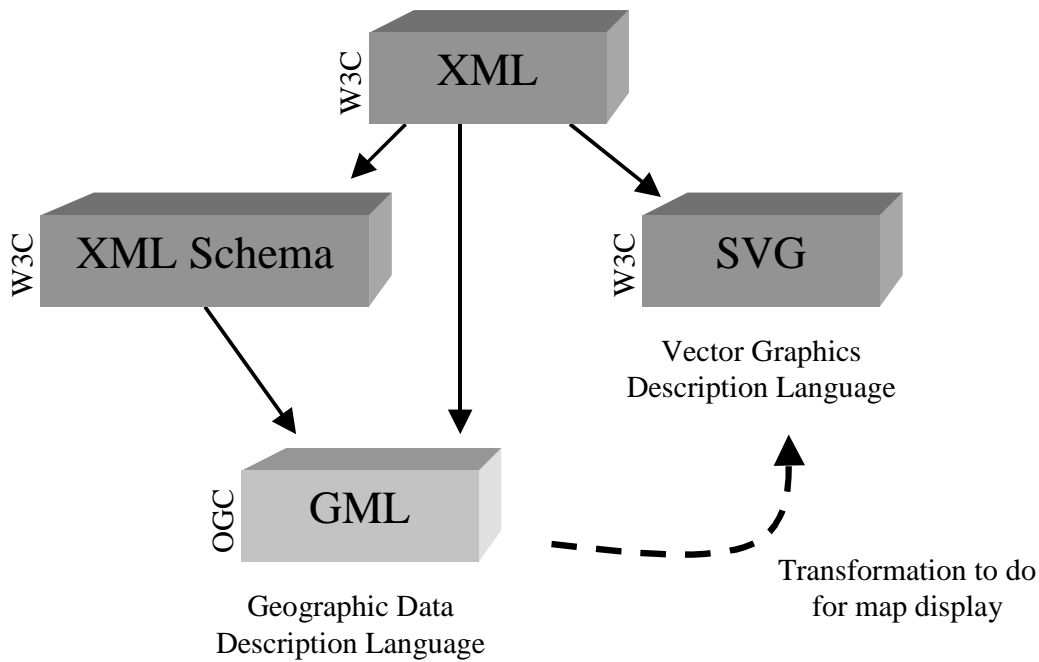


Figure 6.1 : GML et SVG

### Geometric objet Correspondence

The correspondence between the GML geometry type and the SVG structure is easy enough to establish as we retrieve the same geometric figure. Note that a Point will be represented with a rectangle, a circle or a more complex pattern more figurative of the object. Here is a table with the transformations :

GML	SVG
<Point>	To be determined : rectangle, circle, icon at the point
<LineString>	<polyline>
<Box>	<rect>
<LinearRing>	Used with <svg:polygon>
<Polygon>	<polygon>
<MultiPoint>	To be determined : rectangle, circle, icon at the point
<MultiLineString>	A group <g> of <svg:polyline>
<MultiPolygon>	A group <g> of <svg:polygon>

Table 6.1 : Geometric object Correspondence between GML et SVG

The coordinates of geometric elements in a GML dataset are relative to a Spatial Reference System (SRS), given by his name in the 'srsName' attribute. With SVG, the coordinates can be express in pixel, centimetre, inch, ... We need a conversion to superimpose the GML space defined by the 'boundedBy' property with the SVG user space.

### Feature Correspondence

The generated SVG document will have a structure similar to the GML document. We use the 'class' property of graphical objects to identify the element origin. We could use the property 'gml:name' or 'gml:fid' to identify the SVG element with an 'id'.

The properties of one feature are to group with the <g> element. The features collections are to group with the <g> element. With this, we will be able to shown or not a feature group, for example, for a layer of a map. For geometric properties, the SVG element to generate are described in Table 6.1. For non-geometric properties (text, integer, decimal, ...),

it could be interesting to visualise the corresponding text on the map like a legend of the graphic object, the name of the city for, example. A summary table is given in Table 6.2 with transformation models.

GML	SVG
GML Feature  <pre>&lt;Xxx&gt; &lt;gml:name&gt;feature_name&lt;/gml:name&gt; ... &lt;/Xxx&gt;</pre>	SVG group  <pre>&lt;g class="Xxx" id="name"&gt; ... &lt;/g&gt;</pre>
Geometric property : a point in the example  <pre>&lt;Xxx&gt; ... &lt;gml:location&gt;   &lt;gml:Point&gt;     &lt;gml:Coord&gt;       &lt;gml:x&gt;valuex&lt;/gml:x&gt;       &lt;gml:y&gt;valuey&lt;/gml:y&gt;     &lt;/gml:Coord&gt;   &lt;/gml:Point&gt; &lt;/gml:location&gt; ... &lt;/gml:location&gt;</pre>	Base element as in Table 6.1 : a rectangle in the example  <pre>&lt;rect x="valuex" y="valuey" height="10" width="10" class="Xxxlocation"/&gt;</pre>
Non-geometric property : text, integer, decimal  <pre>&lt;gml:name&gt;feature_name&lt;/gml:name&gt;</pre>	Text as legend  <pre>&lt;text x="valuex" y="valuey" class="Xxxname"&gt;feature_name&lt;/text&gt;</pre>

**Table 6.2 : Correspondence between GML Features and SVG elements**

An application example is given in Table 6.3.

GML	SVG
<pre>&lt;WaterTank&gt;   &lt;gml:name&gt;Drueulul&lt;/gml:name&gt;   &lt;gml:location&gt;     &lt;gml:Point&gt;       &lt;gml:Coord&gt;         &lt;gml:x&gt;512.0&lt;/gml:x&gt;         &lt;gml:y&gt;362.0&lt;/gml:y&gt;       &lt;/gml:Coord&gt;     &lt;/gml:Point&gt;   &lt;/gml:location&gt; &lt;/WaterTank&gt;</pre>	<pre>&lt;g class="WaterTank" id="Drueulul"&gt;   &lt;rect y="357" x="507" height="10" width="10" class="WaterTanklocation"/&gt;   &lt;text y="362.0" x="512.0" class="WaterTankname"&gt;Drueulul&lt;/text&gt; &lt;/g&gt;</pre>

**Table 6.3 : Example : A water tank has a name and a Point location**

### Display Style

GML 2.0 doesn't use any graphic specification. A GML document doesn't contain any information on the presentation of the geographic features. The visualisation of a GML document is dependent of the possible vector graphic format. A lot of symbolic representation can be generated with a single GML file ; these different representations can include different graphical format and different symbol. A GML file can generate a lot of map depending on the style applied to the geometric objects.

If we use only correspondence seen before, all the objects will have the same aspect and we can, for example, distinguish a road from a river. In the same way, the texts added on the map as legend must be differentiated.

These indications, absent from a GML document, as it is not its vocation, must be described in an other document.

In SVG, there are three common usage for styling :

- either **directly in the SVG content** ; SVG content can be fully specified without the use of a style sheet language ;
- either **SVG content generated as the output from XSLT transformation** [XSLT10] [XSL10] ; an XSL style sheet can be used to transform XML data extracted from databases into an SVG graphical representation of that data ; a XSLT processor must be used to generate the document generally on server side.
- or **SVG content styled with CSS** [CSS1] [CSS2] ; styles are described in a Cascading Style Sheet (CSS) and the SVG document use them ; to change a style, you only have to change the corresponding entry in the CSS file.

The use of CSS style sheet is flexible enough to change easily the map presentation. Using the class names given to SVG elements, the user can write or generate the styles to each class corresponding to SVG features and properties. This style sheet can described the element to shown or not with the style attribute 'display' and/or 'visibility'.

Example : style sheet "watertank.css"

```
.WaterTank {display:inline}
.WaterTanklocation {fill:red;stroke:blue;stroke-width:2;
visibility:visible}
.WaterTankname {fill:black; visibility:visible}
```

SVG view with the application of the style sheet to the previous example



### Fusion

To realise practically these transformation from GML to SVG in a generic way, we propose to construct a generator with a GML document as input and an SVG document as output. This generator must be help with style sheets (Figure 6.2).

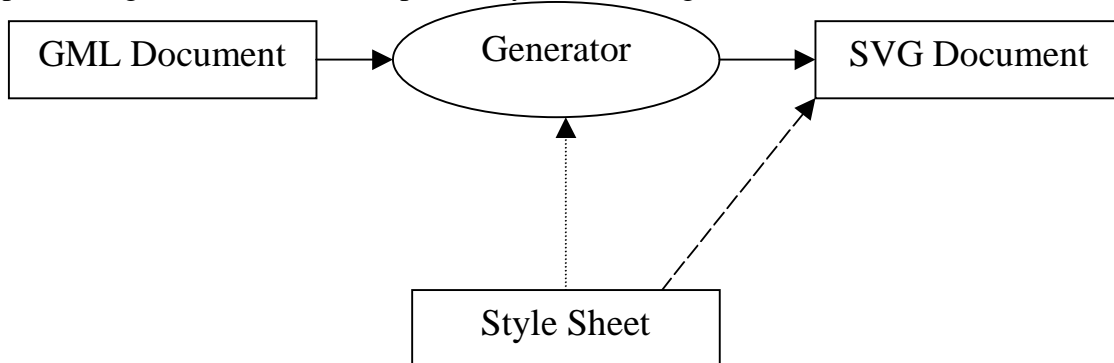


Figure 6.2 : SVG document generation

The SVG generator can be realised on many ways : either by a program that parse the XML file, either by transformation XSL (XSLT). We choose the second solution because it is easy to do, the XSLT was designed specially for that. On the other hand, for special processing like extensions described in the next chapter, it would be necessary to use the first solution. For example, the actual generator doesn't use the style sheet, this could be done to generate only element with a style, the other being useless. Today, the user must specify properties to display and the corresponding elements are generated even they are useless.

XLST transformations to realise can be generic at two levels. First, if the application uses only geometric properties defined in the GML features schema, then, a general XSLT style sheet can be defined and used for all applications of this kind. Second, if the GML

application schema use specific geometric properties, then, the defined XSLT style sheet can be only use with the applications using this schema.

## 7 Conclusions

In this project, we use the latest standards that we found well adapted to the problem. These ideas has been validated on some examples and a prototype is going to be realised with some new functionality. Some SVG capabilities are not used and we'll be useful for the map display. First, the addition of interactivity to manage layers : the base structure of the SVG document has been created in this way, we only have to complete with corresponding buttons and scripts. In the same way, zoom and moving functions can be write with scripts using the DOM [DOM] of SVG. A map is not complete without a legend, so we need the automatic generation of legends. An important problem in cartography is to establish the rules for the positioning of legend on the map to avoid overlapping, superimposing, ...

Links and associations of GML have not be used but we can imagine to visualise the links between the features to show dependencies. For example, a water tank supply water to a village, we can draw a line between the graphical representations, from the tank to the village. The use of the Xlink of GML will permit to establish hyperlink between graphical elements.

The GML schema parsing could add a lot of semantic for the help to design : to develop a style sheet editor which propose features and properties of the schema, to realise a automatic style sheet generator from the schema.

We have to study how to manage temporal series to animate objects on the map.

This spatial data visualisation try to establish a correspondence between descriptive data and graphical data. It is the first step, in the global project to integrate systems with general use of GML as standard format.

## References :

- [CSS1] Cascading Style Sheets, level 1 - W3C Recommendation, 17 Dec 1996, revised 11 Jan 1999, <http://www.w3.org/TR/REC-CSS1>
- [CSS2] Cascading Style Sheets, level 2 Specification - W3C Recommendation, 12 May 1998, <http://www.w3.org/TR/REC-CSS2>
- [DOM] Document Object Model - <http://www.w3.org/DOM>
- [GML20] Geographic Markup Language (GML) 2.0 Specification. OGC recommandation Paper, 20 february 2001, < <http://www.opengis.net/gml/01-029/GML2.html> >
- [GML20f] Spécification de Geographic Markup Language (GML) 2.0. Recommandation de l'OGC - Traduction française de **Gilles Taladoire**, version du 20 février 2001, < <http://www.opengis.net/gml/???> > to be posted
- [ISO211] ISO/TC 211, Geographic information/Geomatics - <http://www.statkart.no/isotc211/scope.htm>
- [OGC] OpenGIS Consortium – <http://www.opengis.org> – <http://www.opengis.net>
- [OGC99] OpenGIS Abstract Specification - <http://www.opengis.org/techno/specs.htm#abstract>
- [SVG] Scalable Vector Graphics (SVG) 1.0 Specification. W3C Candidate Recommendation, 2 November 2000, < <http://www.w3.org/TR/2000/CR-SVG-20001102/index.html> >

- [VRML200x] The Virtual Reality Modeling Language. Draft International Standard ISO/IEC 14772:200x. < <http://www.web3d.org/TaskGroups/x3d/specification> >
- [WFS] Web Feature Server : OpenGIS Project Document 01-023, Editor: Panagiotis A. Vretanos (CubeWerx Inc.) issued by the WWW Mapping SIG - <http://www.opengis.org/techno/discussions/01-023.pdf>
- [XLink] XML Linking Language (XLink) Version 1.0. W3C Proposed Recommendation (20 December 2000). < <http://www.w3.org/TR/xlink> >
- [XML10] "Extensible Markup Language (XML) 1.0", February 1998, < <http://www.w3.org/TR/REC-xml> >
- [XMLNS] Namespaces in XML. W3C Recommendation (14 January 1999). < <http://www.w3.org/TR/REC-xml-names> >
- [XMLSchema-1] XML Schema Part 1: Structures. W3C Candidate Recommendation (24 October 2000). < <http://www.w3.org/TR/xmlschema-1> >
- [XMLSchema-2] XML Schema Part 2: Datatypes. W3C Candidate Recommendation (24 October 2000). < <http://www.w3.org/TR/xmlschema-2> >
- [XPointer] XML Pointer Language (XPointer) Version 1.0. W3C Candidate Recommendation (7 June 2000). < <http://www.w3.org/TR/xptr> >
- [XSL10] Extensible Stylesheet Language – W3C Candidate recommendation <http://www.w3.org/TR/xsl/>
- [XSLT10] Extensible Stylesheet Language – W3C Recommendation <http://www.w3.org/TR/xslt.html>