Geologic Data Transfer Using XML

Simon J.D. Cox
CSIRO Exploration & Mining
PO Box 437, Nedlands, WA  6009
Australia


Abstract

XML is a popular standard for structured-data transfer using plain text. The tag-set and structures must be defined for each particular application, such as geology. Developing the tag-set thus requires design of an application-domain data-model, which is then serialised as a schema document. A general data-model for geospatial information has been developed and standardised through ISO. The feature model suits a database-oriented datastore, in contrast to the conventional map-oriented "coverage" approach. An XML encoding of the model, the Geography Markup Language (GML), has been developed through the OpenGIS Consortium's (OGC) implementation program. GML provides generic components for the description of "features", including geometry and patterns for building application-specific feature-types. We are developing the eXploration and Mining Markup Language as an ISO compatible XML encoding for geologic data by making it an "application" of GML.

Web technology

New computing platforms

The emergence and popularity of the World Wide Web has led to rapid developments in computing platforms. Important components that have been particularly affected include:

- **user interfaces** – the browser is now used to host the interface of many new applications;
- **client-server architectures** – stateless communication over http is the dominant low level protocol;
- **data representations** – information is carried in “markup”, and interpreted by the client.

The new tools are finding application in traditional desktop settings as well as in web-hosted processing. Such uniform use of the internet protocols reinforces the fact that the resources that are available to a user are no longer restricted to what is on their local disk or workplace LAN, but extend across the internet.

In this environment, developers working in a particular technical domain can plan to leverage the platform by focussing their effort at a higher level, stitching together generic components, and only doing low-level development where domain specific expertise is needed. For example, back-end interfaces to databases are standardised, and the application of access restrictions to information, perhaps involving financial transaction processing, are generic e-business components for which standard tools and products can be used. But the content of the information that is transported between services (datastore, processing application, etc) must embody the
information model and support the processing requirements (“business rules”) that characterise the domain.

**Data representation - markup languages**

The simplicity of the webpage format HyperText Markup Language [HTML] is the feature that was perhaps most important in facilitating the rapid growth of the World Wide Web. HTML provided page description together with embedded hyperlinks and reference to images (Example below: Simple webpage). It is text-based, and can thus be created and modified using the most basic tools. It is compact, with the most complex pages typically an order of magnitude smaller than equivalent word-processor documents. The details of the appearance of the document are largely delegated to the client (usually a web browser), which “processes” the data to produce a formatted version for the user to read.

```html
<html>
<head>
<title>Sample report</title>
</head>
<body>
<h1>Sample Report</h1>
<img align="right" src="mugshot.jpg">
<h2><i>J. Q. Geologist</i></h2>
<h3>Huge Resources
PO Box 999
Digemup
WA
</h3>
<p><i>Created:</i> 1999-11-08<br><i>Last Modified:</i> 1999-12-01</p>
<hr>
<img align="right" src="locmap.gif">
<p>We report the results of a drilling program. The <a href="log.dbf">drill-log</a> database and <a href="chem.xls">assay</a> results are available separately. </p>
<p>Etc ...</p>
</body>
</html>
```

**Simple webpage with HTML presentation markup**

The markup strategy used in HTML – in which “tags” that label structure and function are embedded within data recorded in plain text – was derived from Standard Generalized Markup Language [SGML]. However, only a limited set of object types and behaviours are built in to HTML (headers, paragraphs, character-styles, tables, etc), and it is not extensible. Even though the layout of information on a page can usually convey quite sophisticated semantics to a human reader, data delivered in HTML cannot be accessed directly by anything other than web-page processing (e.g. rendering) software.
Full SGML allows the user to define custom tags and structure for document-oriented data, and has been used to develop specialised text management and delivery systems. The promise of using markup for more general data on the web led the World Wide Web Consortium [W3C] to develop a “streamlined” version of SGML, the eXtensible Markup Language [XML]. The tags or “element names” are usually natural language words, which make the data instance largely self-contained and effectively self-describing (examples below: webpage, structured data). The tags are delimited from the data by being enclosed in &lt;, and other data is stored as labelled “attributes” within the tags.

Many resources introducing XML are available, so no further explanation will be included here.

Simple webpage with XML semantic markup
Structured information in XML

The basic XML specification [XML1] was designed to be simple and short, by removing many optional features present in SGML. However, the application flexibility of XML has spawned many supporting technologies [XML] (e.g. XSL and XSLT for formatting and transformations, DTD and XSD for structure and validation, XLink, XPath and XPointer for references, XML Query, XInclude, etc.). The development of the specifications for technologies in the XML “family” currently comprises the main activity of W3C. XML has also now almost completely supplanted SGML in even its traditional publishing domains. Processing libraries for XML are widely available, based on standard mappings from XML documents or streams to object models (DOM and SAX). Parsers and processors are increasingly built in to standard software development environments.

Data structure - schema languages

Strictly speaking, XML is a meta-language, used for the construction of specific languages for the serialisation of structured data for particular ends. The underlying data model is a tree, defined by the pattern of nesting of the tags. But the tag-set, and the actual structures in which they are nested, must be defined for each particular application, such as geology. This results in a specialised language with syntax from XML, but vocabulary and structure to match the application domain. Hundreds of XML based languages have been published or announced publicly (see Cover Pages [Cover] for a good summary) and many more have undoubtedly been developed for local use. Of particular interest in geology are MathML for the representation of math data for both presentation and symbolic manipulation, Scalable Vector Graphics [SVG] and eXtensible 3D [X3D] for visualisation.
The tag-set and structure of an instance in one of the XML languages is described formally in a *schema definition*. The most basic use of a schema is to validate data, ensuring that it has all the right pieces in the right places. A number of schema definition languages have been developed. The Document Type Definition (DTD), inherited from SGML and described in the XML 1.0 specification [XML1], uses a grammatical approach, and is most suitable for (semi-structured) textual information. At the other extreme, W3C XML Schema Definition Language [XSD] is object-oriented, and is most suitable for (strongly typed, fully structured) numeric and tabular data. Other alternative schema languages such as Relax and Schematron retain some of the advantages of XSD (e.g. a set of basic data types such as text, number and date; the schema definition is itself an XML document) while avoiding some of the structural complexity for applications where this is not needed.

XSD is particularly interesting, since it gives a *data model* packaged in a form that can be used directly, for data validation and interface construction at least, in widely available and inexpensive software with no additional steps (feature model, sample model, borehole model).

**Modelling the real world**

**Geospatial Feature Model**

Whichever schema representation tool is to be used, in order to develop the tag-set it is necessary first to analyse the requirements of the application-domain, and describe those in a data-model.

Geological data is used for a variety of activities, with the conventional representation of choice varying accordingly. For example, in regional exploration, remote-sensing imagery, geophysics, and compilation maps are important, while for ore-deposit delineation, drill-holes and assays located in 3-D are critical. The detailed data models for these different data have some significant differences. However, they have a key feature in common: the objects of interest are positioned in, or relate to geospatial locations. Because of this, Geographic Information Systems (GIS) software has been a valuable tool in storing and analysing geological data.

A data-model intended to provide the basis for future representations of geographic information has been developed and standardised through ISO Technical Committee 211 [TC211]. This is described in a series of more than 20 specifications in the series ISO 191##, most currently in their second or third “Committee Draft” or in “Draft International Standard” status. ISO is supported by many statutory organizations.

The modelling uses an object-oriented approach. The basic component – an identified object with a geospatial context – is called a "feature". This includes not only classical objects such as "fault", "sample" or "reservoir", but also extended or compound features like "image", "survey" and "model". The feature model suits a granular datastore, in contrast to the conventional map-oriented "coverage" approach.

TC 211 primarily focusses on the development of general models described in *Abstract Specifications*. The models are documented with class-diagrams using the graphical Unified Modelling Language [UML] (feature model). Again, these abstract
specifications merely provide a framework or tool for the development of models for specific domains, called *Application Schemas*. A key component of each Application Schema is a *Feature Catalogue* or list of definitions of named feature types and their attributes, which are useful in the particular application domain.

Complementing the “top down” approach from ISO/TC 211, the OpenGIS Consortium [OGC] is a developer- and vendor-led effort to define open standards for GIS software interoperability. OGC activities are now mainly organised around the *Interoperability Program* in which groups of developers, normally drawn from several different organizations, produce prototype applications for “Testbeds” based around particular scenarios. The technologies used in these experiments are then formalised into *Implementation Specifications*. All the significant GIS and database vendors work with OGC, as well as a host of smaller integrators and service providers.

Notable OGC successes include the Web Mapping Testbed, which led to the *Web Map Server* [WMS] specification, which defines how to request and receive a specified map in the form of an image, and the extension of this to the Web Feature Service (WFS), which defines how to request the description of a particular Feature or Feature Collection from a service. Other specifications [OGCspecs] either completed or under development by OGC include SQL, CORBA, and COM/OLE implementations of the Feature Model, catalog services, coverage/grid/image, coordinate transformations, geocoding, geoparsing and geospatial fusion services (GFS). Some of these specs cover components of “classic” GIS, though using technologies that can be applied to a web context. Others augment this with links to other data types and services. For example, GFS concerns the automatic assignment of unstructured data (e.g. webpages) to a geographic context.

For the WFS, the Feature data is returned as an XML document, using the *Geography Markup Language* [GML2], which is defined in an XML Schema based on the ISO Feature Model.
Geographic information in XML

So this is how the loop gets completed. ISO/TC 211 has developed a *model* for describing geographic information as *features*. The feature model is documented in UML. W3C have developed XML, which is a method for serialising structured *data* as a *text document*. The structure of an XML instance is defined in a *schema*, which is the serialisation of a model. And OGC have taken the ISO model for geographic features, and produced a corresponding XML Schema, GML, which thus defines the structure of an XML document containing geographic information. Datasets encoded using GML can then be used in transactions between web-based geospatial services, or even on between desktop applications.

GML provides

1. generic components, such as geometry (*GML geometry properties*), used in the description of "features",
2. patterns for building application-specific feature-types;
3. very flexible data normalisation through use of xlinks to previously encoded elements.

<table>
<thead>
<tr>
<th>Geometry components built-in to GML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
</tr>
<tr>
<td>LineString</td>
</tr>
<tr>
<td>LinearRing</td>
</tr>
<tr>
<td>Polygon</td>
</tr>
<tr>
<td>Box</td>
</tr>
<tr>
<td>Triangle</td>
</tr>
<tr>
<td>_CurveSegment</td>
</tr>
<tr>
<td>ArcString</td>
</tr>
<tr>
<td>Arc</td>
</tr>
<tr>
<td>Circle</td>
</tr>
<tr>
<td>Rectangle</td>
</tr>
<tr>
<td>Knot</td>
</tr>
<tr>
<td>BSpline</td>
</tr>
<tr>
<td>Bezier</td>
</tr>
<tr>
<td>MultiGeometry</td>
</tr>
<tr>
<td>MultiPoint</td>
</tr>
<tr>
<td>MultiLineString</td>
</tr>
<tr>
<td>MultiPolygon</td>
</tr>
<tr>
<td>MultiTriangle</td>
</tr>
<tr>
<td>MultiCurveSegment</td>
</tr>
<tr>
<td>MultiArcString</td>
</tr>
<tr>
<td>MultiArc</td>
</tr>
<tr>
<td>MultiCircle</td>
</tr>
<tr>
<td>MultiBSpline</td>
</tr>
<tr>
<td>MultiBezier</td>
</tr>
</tbody>
</table>

GML cannot be used directly as there are no non-abstract feature types (*GML Feature*). For a particular application domain, the Feature Catalogue from its abstract Application Schema is implemented in XML by developing an XML Schema that imports the components from GML and uses its patterns in a set of XML element declarations for the feature types in the domain.
In practice this is done using XML schema types that extend the AbstractFeatureType from GML by adding the appropriate geometric and non-geometric properties for the feature type. For example, a Sample might have a location and set of observations, while a Borehole might have a collarLocation, holePath, a set of logs, and optionally some individual samples (examples below).

Thus, geology data types that can be modelled as (UML) specialisations of ISO features, can be encoded in XML using an application schema that extends GML.
XML “Sample” schema fragment

An XML “Sample”
XMML “Borehole” model
<xsd:element name="Drillhole" type="xmml:DrillholeType" substitutionGroup="gml:_Feature"/>
<xsd:element name="Borehole" type="xmml:DrillholeType" substitutionGroup="xmml:Drillhole"/>
<xsd:element name="Well" type="xmml:DrillholeType" substitutionGroup="xmml:Drillhole"/>

<xsd:complexType name="DrillholeType">
  <xsd:annotation>
    <xsd:documentation>A drillhole is a Feature with a collarLocation, zero or more drillHoleShape's, zero or more drillHoleLog's, and zero or more related features</xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element ref="md:metadata" minOccurs="0"/>
        <xsd:element ref="xmml:collarLocation"/>
        <xsd:element ref="xmml:holeShape" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="xmml:holeLog"/>
          <xsd:element ref="xmml:relatedFeature"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

XMML “Borehole” schema fragment
Customisation for Geology

*eXploration & Mining Markup Language*

In a project coordinated by CSIRO Exploration and Mining, we are developing a representation for geology called the *eXploration and Mining Markup Language* [XMML]. The project has the support or involvement of several companies involved in the resources industry and a large number of Geological Surveys (for an up-to-date list, please consult the website). We are also liaising with several prominent industry software vendors.

There have been previous attempts to develop data models or transfer formats for geological exploration data, in both public and proprietary settings, none of which have had the widespread impact that was hoped for. This was at least partly due to
the fact that there was no effective consensus regarding the tools, and the available infrastructure did not encourage open standards. In contrast, we now have both an effective and widely available tool (XML) and a platform that rewards the use of open standards (the web). The remaining challenges are (1) to accomplish the technical work, and (2) to encourage community consensus on adopting this uniform approach.

Several of the earlier projects provide useful analyses, summaries or even models that can contribute to the present work. Of particular importance are industry standards for samples and assay data, statutory reporting standards for samples and borehole data, statutory metadata schemas, and industry standards for geophysics data. In both modeling and XML development we also draw on the experience of the Petrotechnical Open Software Corporation [POSC].

XMML will be delivered as a set of XML Schema documents that define the xmm1 namespace. The schemas will be publicly available which will enable unrestricted uptake. The XMML schemas <import> the GML schemas in the gml namespace (currently: geometry.xsd and feature.xsd) and possibly other useful schemas, such as W3C type libraries [XMLtypelib], POSC units-of-measure definitions [UoM], as appropriate. The XMML schemas will primarily comprise a catalogue of feature type definitions, which can be used directly. However, it will also be possible for developers to <import> the definitions from XMML in the construction of other schemas in other namespaces.

In conjunction with the Schemas, we will distribute several XSLT stylesheet documents [XSLT], for conversion of data stored in XMML documents into a variety of file formats useful for other purposes: text formats for import into legacy software, SVG and X3D graphics, and HTML for reports. We also plan to set up online validation and conversion services.

Version 1 of XMML is scheduled for delivery at the end of 2001. It will include (at least) samples, boreholes, geophysical surveys, some structural geology objects, and methods for encapsulating textual reports. These definitions will be refined, and more feature types added in continuing work. A significant advantage of XML and the “tree” data model is that it is easily extensible, so can be delivered in working form only covering part of the domain, with the additional components added later, in a way that does not invalidate data instances that conformed to the earlier version.

Technical documentation of the models and schemas, and example datasets conforming to the current versions of the schemas, are provided on the XMML website [XMML], which should be consulted for more detail.

**Alternative models – early vs. late-binding**

The classic approach in GIS implementations has been to treat geometry as primary, with attributes or properties associated with each geometric object. Classification of objects into types was done
• implicitly, through the object being part of a layer or a member of a “coverage”;
• explicitly, using one of the attributes to store a label;
• at run time, by computing some function of the stored attribute values.

The North American Data Model for geologic maps [NADM], under development through USGS, GSC and the state and provincial surveys, goes further and overlays a very explicit classification model. Three broad classes of objects are recognised and stored:

1. spatial items;
2. properties, which are observations made on the instances; and
3. concepts, which are model-based classifications.

Geologic entities are created over the top of these objects, by asserting relationships between objects from the different groups. The semantics of an entity is thus primarily embodied in the relationships, and can be changed by switching the latter without having to delete or create any new primary objects in the database. For example, the interpretation of an entity represented as a line on a map may be changed from an unconformity to a thrust fault as a result of a new observation being made on the object. The classification may also be modified in light of later conceptual developments.

The NADM thus supports a late binding, weakly typed view of the world. The atomic observations are primary, and everything else derived. The NADM classification model supports a scholarly approach, which is primarily useful for interpretative geology. It should be noted that the principal use-case for NADM is for a datastore from which (2D) compilation maps can be generated.

In contrast, in the Feature Model the set of properties that are associated with an object is fixed, determined by its feature type. It is necessary to assign each object to a feature type or class prior to storage. Geometry is merely one of the several properties of an object, rather than its primary identity.

The Feature Model is thus an early binding or strongly typed view of the world. The set of required observations is model driven, determined by the initial classification of each feature. While this is straightforward for engineering objects, such as boreholes or declines, the representation of interpretative geology is necessarily frozen in time. Instantiation of a Feature may be considered to be a (denormalised) snapshot for transfer between model-aware applications.

Nevertheless, it would be possible to produce a Feature Model representation of NADM (sketch below), by defining finer-grained features, corresponding to members of the NADM classes. The higher-level NADM entities would be built as Features that only carry relationships, encoded in GML using the xlink capability.
Uses for XMML

In developing XMML we have clear but limited goals: it is primarily a piece of infrastructure, which if widely adopted will relieve applications developers of many burdens associated with data import and export. XMML is compatible with current and emerging web transaction standards, such as SOAP and Microsoft.NET, but will operate deep within an applications environment and should not be encountered directly by users.

The XMML data model, which is derived from the ISO Feature Model, is intended to be a neutral model for the information, and may not be appropriate for particular processing and analysis operations. It is not based on any legacy systems, and is primarily a transfer model. Applications software will have to map the objects implied by the XMML structure into objects suitable for the application. In some cases it may be convenient to do this external to the application, prior to loading, using transformation software such as XSLT processing.

For a relatively trivial example, a generic viewer (such as the one freely available from Adobe [SVGview]) can be used to inspect a dataset if the XMML is transformed to SVG. The transformation might use an XSLT script for the desired mapping from feature properties to graphical attributes such as colour, line-thickness, pattern, labelling, etc. In this case, the XSLT rules encapsulate a legend for the visualisation.

Since XML is a self-describing clear-text file-format, XMML is also a prime candidate for archival use. This is an important motivation for the involvement of statutory agencies.
The additional advantages of basing an XML encoding for geology on GML are that by building on generic geospatial technologies, we confine our effort to geology domain specific components, while maximising our potential use of generic DBMS, GIS, CAD and graphics software and enabling us to also use the emerging web-services model which is being adopted for OGC applications.

OGC services model (from WMS)

References


[HTML] HyperText Markup Language Home Page http://www.w3.org/MarkUp/


[SOA] Simple Object Access Protocol (SOAP) 1.1. W3C Note 08 May 2000 Authors: Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, Dave Winer http://www.w3.org/TR/SOAP/


[UoM] XML Patterns and Objects – Units of Measure, POSC. http://www.posc.org/ebiz/pfxml/patternsobjects.html#units


[X3D] eXtensible 3D http://www.web3d.org/x3d.html

[XML] eXtensible Markup Language http://www.w3.org/XML/


[XMLtypelib] XML Schema Type Library http://www.w3.org/2001/03/XMLSchema/TypeLibrary.xsd

IAMG 2001