# Advances in GML for Geospatial Applications

Chang-Tien Lu, Raimundo F. Dos Santos Jr, Lakshmi N. Sripada, Yufeng Kou
Department of Computer Science
Virginia Polytechnic Institute and State University
*{ctlu,rdossant, lsripada,ykou}@vt.edu*

## *Abstract*

*This paper presents a study of Geography Markup Language (GML)*[1]*, the issues that arise from using GML for spatial applications, including storage, parsing, querying and visualization, as well as the use of GML for mobile devices and web services. GML is a modeling language developed by the Open Geospatial Consortium (OGC) as a medium of uniform geographic data storage and exchange among diverse applications. Many new XML-based languages are being developed as open standards in various areas of application. It would be beneficial to integrate such languages with GML during the developmental stages, taking full advantage of a non-proprietary universal standard. As GML is a relatively new language still in development, data processing techniques need to be refined further in order for GML to become a more efficient medium for geospatial applications.*

## 1. Introduction

With the increase in the use of the Internet as a medium for information exchange, there has also arisen the need to develop applications that exchange data seamlessly. eXtensible Markup Language (XML) is an attempt in this direction. As a meta language, other languages can extend XML for use in specific areas of application. Geography Markup Language (GML) is an XML encoding designed for use with geographic information. This language helps in the storage, exchange, and modeling of geographic information containing both spatial and non-spatial attributes. GML uses the concepts provided in the Abstract Specification of the Open Geospatial Consortium (OGC) for modeling geographic objects, such as geometry, topology, and features. Support for more complex objects has been incorporated in GML 3 [5], whereas previous versions

[1] Please refer to Appendix A for a list of acronyms used in this paper.

only accounted for simple features. GML data is self-descriptive, serving as a mechanism for information discovery, retrieval and exchange [35].

A geospatial application is supported by a database or file system that can handle spatial data types. Spatial data objects not only have a non-spatial description, such as name and population, but also have spatial attributes, such as location, geometry and neighborhood properties. A geospatial application must provide various functionalities, including input, storage, retrieval, selection, analysis, and display of the information [38]. Although these features are also provided by traditional applications, they seldom handle spatial information in a uniform format, which may lead to problems in the exchange of spatial information. GML is designed for use as a common language that applications can use to communicate with each other and exchange information with minimal overhead. GML representation of information is unique, the way its information is used can differ, and its meaning can vary according to context.

Using GML for geospatial applications has both advantages and disadvantages. GML documents nest spatial data types, permitting the effective representation of the various components of spatial data. This data has to be stored in such a way as to allow efficient query processing. However, extracting information from GML documents can be challenging due to time constraints and application complexity. The choice of spatial XML RDBMS (Relational Database Management System) also plays a role in the data extraction process. Inefficient query processing, especially for large data sets, is often difficult to overcome. Since GML is based on XML, the query languages and other data processing capabilities available to XML can also be used for GML. However, ideally, they should be extended to support the processing of spatial and temporal data that makes GML powerful. Although GML is a promising language, mechanisms to maximize its full potential have not yet been fully developed.

This paper addresses various aspects of using GML for geospatial applications. It opens with a discussion of storage of GML documents and provides a comparison of different database types that can be used for GML. This discussion includes GML schemas as well as the advantages and disadvantages of handling GML data in various formats. We then go on to present an overview of parsers and query languages, as well as list indexing approaches that can be applied to GML documents. The usability of standard XML tools when applied to GML applications is also considered. For example, GML visualization typically relies on commercially-available viewers, with Scalable Vector Graphics (SVG) viewers being the most common. We weigh the pros and cons of implementing tools designed specifically for GML. In addition, this paper discusses how GML can benefit from common technologies, such as SVG, and how they can facilitate visualization. Because geospatial applications are now being utilized in mobile environments, we devote part of the discussion to the use of GML in portable devices. Similarly, we include Web Services as a growing field of geospatial application, and provide an overview of common Web Services solutions. The goal is to help understand some of the robust aspects of GML, identify some of its not-so-desirable characteristics, and list current research issues.

This paper is organized as follows: In Section 2, we discuss GML storage, schemas, and the differences between GML versions. GML parsers and query languages are addressed in Section 3. Techniques for data transformation and visualization of GML documents are presented in Section 4. Applications of GML for mobile devices as well as GML Web Services are shown in Section 5. Finally, we list various GML-related research issues in Section 6, and sum up the paper's conclusion in Section 7.

## 2. GML Schemas and Storage

GML targets both information storage and retrieval in its specifications. While this standardization process brings benefits, it has also created issues. To date, GML has not yet become the most efficient medium for dealing with geospatial data over distributed systems, although it clearly has the potential to do so. Prior to the initiatives proposed by the OGC, many vendors had already created their own proprietary data storage and retrieval strategies, many of which worked efficiently in their application realm. Systems were not always based on XML grammar, and the ones that did use XML utilized their own set of rules to translate incoming data and process them.  It is arguably true, however,  that in adapting to OGC standards, many institutions realized a tradeoff between performance and interoperability. GML 3 is much heavier in functionality than its predecessors, but the extra baggage is necessary in order to address the robust needs of today's applications.

Spatial data is heavy by nature. Any single map, for instance, may consist of millions of attributes, limited only by how much information is made available to the outside world. As a consequence, GML documents can be (and often are) very large, raising concerns about processing and transport. The GML specification does not address (nor does it intend to) functional aspects related to performance. Users must be aware that the burden of efficient processing, as well as that of transmission times, lies solely on their shoulders. While GML can be used for storage of geospatial data in plain text files or XML databases, this is not always the case. In fact, many institutions implement their own storage strategies in different ways. In part, storage is dictated by several factors, such as the availability of computing resources, the ability to migrate from legacy to newer systems, and funding. However, GML documents were designed to be transferred between systems in a transactional fashion, allowing users to process the incoming data as it streams in.

## 2.1. GML Schemas

The method of generating GML documents is irrelevant, and  they can be created by any available document generation tool. All that is required is that the documents conform to the requirements in the GML document specification. GML version 1 used Document Type Descriptors (DTD) on Resource Description Framework (RDF) Schemas to define elements and their attributes. However, DTDs have some disadvantages. For instance, they are not written in XML, which makes them inconvenient to interpret. GML 2 removed DTDs and RDF Schemas, and GML 3 extends the use of XML even further. A GML schema is an XML Schema, which means that a single interpreter can be used for both the schema itself and the GML document. A GML application schema restricts and extends GML definitions and serves to enforce the accurate construction of application-specific GML documents [42].

GML defines various XML schema types and elements such as features, geometries, and topologies through a hierarchy of GML objects. The GML specification provides a series of schemas for describing geographic data in XML. The GML objects defined in the OGC specification are broken down into several schema documents that cover aspects such as Feature, Geometry, Topology, Value, Coverage, Temporal, Coordinate Reference System, XLink and StyleDescriptor. Other types represent subtypes of Feature, namely Observation, Coverage and Definition. However, these schema documents do not provide a suitable schema for all the instance documents and only make available the foundation structures that an “application schema” can use. In turn, the application schema applies the relevant features and types needed for the specific domain in question. Depending on the requirements of the application domain, designers can create different types of schemas by extending or restricting the features from the GML base schema. This provides designers with considerable flexibility in using GML to represent a diverse range of spatial objects. There have been initiatives towards the implementation of standard application schemas to specific domains. Brodaric et al., for instance, describe the GeoSciML project as a tailored GML Schema used to manage scientifc data suited for geological mapping [14]. GeoSciML illustrates a strong application of how GML can be leveraged to describe features related to the geological domain, such as Earth structures, fossils, material compounds, and their relevant attributes.

Examples of GML documents and schema documents can be found in the OGC GML specification [18]. GML uses namespaces to distinguish between components with the same name that are defined in different schemas. For example, a component associated with both the *xlink* or *GML namespaces* is defined in the two namespaces as "http://www.w3.org/1999/xlink" and "http://www.opengis.net/gml," respectively. Fragments of GML instance documents are provided in examples 1, 2, 3 and 4. The example code is taken from [18]. Example 1 shows how the namespace declarations associate prefixes used in the instance document with namespace URLs, as in lines 3 to 7. The GML schema location is specified in line 8.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <ex:RoadInfrastructure>
3. xmlns:ex=http://www.opengis.net/examples
4. xmlns=http://www.opengis.net/gml
5. xmlns:gml=http://www.opengis.net/gml
6. xmlns:xlink=http://www.w3.org/1999/xlink
7. xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
8. xsi:schemaLocation="exampleRoad.xsd">
...
14. <featureMember>
15. <ex:Road gml:id="r1">
16. <curveProperty>
17. <CompositeCurve srsName="somelistofcrs.xml#1234">
...
22.   <description>
      The geometry of the road uses different
       interpolations   </description>
23. <boundedBy>
24. <Envelope srsName="somelistofcrs.xml#1234">
25. <pos>0 0</pos>
26. <pos>50 50</pos>
27. </Envelope>
28. </boundedBy>
```

**Example 1: A GML instance document "exampleRoad.xml" [18].**

The OGC Abstract Specification describes a real-world phenomenon in terms of a set of features which may or may not have geometric properties. A spatial feature may be associated with one or more geographic properties, such as location. The feature in Example 1 is *RoadInfrastructure*. A feature is described by a set of properties. For example, curveProperty, as seen in Example 2, is a property of roadInfrastructure.

The values of coordinates for feature geometry are always associated with a Spatial Reference system (SRS), which is a means of referencing geographic features to a specific surface such as that of the Earth [26]. Different SRS have different coordinate values for the same location, therefore all coordinate values must specify which SRS they are using. The SRS for *RoadInfrastructure* is "*somelistofcrs.xml#1234*," for example.

Properties are distinguished from instances by notation: element names corresponding to instances of GML classes start with an uppercase letter (e.g., Curve), while property tags start with a lowercase letter. If a feature has a property that takes a geometry element as its value, it is referred to as a geometry property.

```
<featureMember>
        <ex:Road gml:id="r1">
        <curveProperty>
         <CompositeCurve srsName="somelistofcrs.xml#1234">
```

**Example 2: Feature specification in a GML document [18].**

The term *Envelope* describes a region bounded by a pair of positions denoted by its corners. The "*coord*" type has been deprecated with GML3.0, but included to provide backward compatibility with GML 2. The "*pos*" and "*posList*" elements are now used instead.

```
<description> The geometry of the road uses different interpolations .</description>
 <boundedBy>
   <Envelope srsName="somelistofcrs.xml#1234">
        <pos>0 0</pos>
        <pos>50 50</pos>
     </Envelope>
</boundedBy>
```

**Example 3: Envelope element in GML documents [18].**

*CurveMember* is a property of the *CompositeCurve* object. As in example 4, its children may have geometries with similar or different interpolations. The two curve members in the example are *Curve gml:id "c101" and gml:id "c102."* Some of the elements that can be used as a value of *gml:curveMember* are *gml:LineStringSegment* and *gml:CubicSpline*. These elements are new to GML version 3. Some of the other geometry elements supported by GML are *Point, Linestring, LinearRing, Polygon, MultiPoint, MultiLineString, GeometryCollection* and *MultiPolygon*. A *LineString* is a type of curve that is composed of single segments. It is defined by two or more coordinate tuples, with linear interpolation between them.

```
<curveMember>
    <Curve gml:id="c101">
        <segments>
            <LineStringSegment>
                <coordinates>...</coordinates>
            </LineStringSegment>
            <CubicSpline>
                <coordinates>...</coordinates>
                <vectorAtStart>1 0</vectorAtStart>
                <vectorAtEnd>1 0</vectorAtEnd>
            </CubicSpline>
        </segments>
    </Curve>
</curveMember>
<curveMember>
    <Curve gml:id="c102">
        <segments>
            <CubicSpline>
                <coordinates>...</coordinates>
                <vectorAtStart>1 0</vectorAtStart>
                <vectorAtEnd>1 0</vectorAtEnd>
            </CubicSpline>
            <LineStringSegment>
                <coordinates>...</coordinates>
            </LineStringSegment>
        </segments>
    </Curve>
</curveMember>
```

**Example 4: The main body of GML instance document "exampleRoad.xml" [18].**

By using a few core schemas as defined in the GML specification, new data types can be defined. Further details about GML standards, examples of GML documents, and GML schemas can be found in the GML specification document [18].

## 2.2. GML Storage

Efficient exchange and storage of GML documents is an important issue. GML 3 is substantially more functional than its predecessors, and thus richer in its hierarchy structure. However, data does not have to remain in GML format, but can be stored in an existing database format and converted into GML format whenever needed. GML documents tend to be larger in size than other documents containing the same information. One possible solution to the file size issue is the use of compression. The Gzip format, for example, has been used to reduce GML file size [36], and can result in files as small as the original compressed binary encoding.

GML helps build multidirectional associations among different features and feature properties using XLink and XML pointer language (Xpointer). The data can thus be seamlessly integrated with other GML documents providing different information about the same spatial data. Since GML is a text-based language, it also provides an efficient means for archiving geospatial data, making it unlikely that future software will be incompatible with it [9].

There are several alternative approaches that can be deployed for the storage of semi-structured data or XML documents [25], and Table 1 summarizes some of the most common:

| | **Advantages** | **Drawbacks** |
|---|---|---|
| Object Oriented DBMS | - Some support for GML extracts and loads<br>- Handles complex, inter-related data<br>- Fewer join operations | - Complex model<br>- Difficult to change schema<br>- Language dependent |
| Object Relational DBMS | - Support for abstract data types (ADT)<br>- Relational and object-oriented features<br>- Handling of large objects common in GML<br>- Code reuse | - Difficulty in translating object data to relational data<br>- Comparatively less interoperability than relational DBs<br>- Lack of standards |
| Relational DBMS | - Easy searching<br>- Available indexing support<br>- Widespread use | - Difficult to normalize GML into tables<br>- Complex to extract data into GML |
| XML Database | - Standard XML APIs and tools<br>- Support for SAX and DOM<br>- Support for XSLT and XQuery | - Less SQL support<br>- Newer technology<br>- Less expertise by software developers |
| GML file on disk | - Strong validation<br>- Accepted standards<br>- Little initial processing<br>- Freedom of formats<br>- Less strict rules | - Complex parsing<br>- High storage requirements<br>- Less interoperability between systems<br>- Lack of standards<br>- Little semantic meaning |

**Table 1: Various approaches to storing GML/XML documents.**

In specialized data management systems, such as Rufus [30], Lore [31]and Strudel [21], the models are customized to store and retrieve semi-structured XML data. Storing of GML data can use one of several approaches: a relational model, an object-oriented model, an object-relational model, a specialized XML database, or full file storage on disk. Storing the whole file represents less overhead, since no heavy processing needs to be performed. However, this approach tends to become space-intensive over time. In the case of a relational model, the data is mapped into relations and queries are posted in a semi-structured query language, which is then translated to SQL queries. Using a database designed for semi-structured data seems to be the best approach with respect to scalability and handling large amounts of data. The object-oriented approach is suitable for more complex data, but not necessarily for larger data sets. The relational model is a widely-accepted approach and, if adopted, would help in the future integration of XML databases with other databases. Oracle DBMS and XML DB are examples that support different approaches [6]. Nonetheless, each approach must still be evaluated further in terms of the efficiency of its query processing for large amounts of spatial and non-spatial data. For instance, the object-oriented approach is generally not efficient for queries on large databases. The relational model provides processing advantages due to its availability of ready-made data management tools. However, mapping a GML Application Schema to a relational database tends to result in complex structures (e.g., many tables and relationships), which may degrade system performance. This process needs to be very fine-tuned to be suitable for processing GML data.

The approaches to designing database schemas for XML documents can conveniently be divided into two categories: structure-mapping and model-mapping [48]. Under structure-mapping, the design of the database schema is based on the DTD (Document Type Descriptor), or GML schema that describes the structure of the GML documents. With the model-mapping approach, a fixed database schema is used to store any GML documents without the assistance of GML schema or DTD. These mappings are performed on element types, attributes and text.

Corcoles and Gonzalez compare three types of document-storing techniques based on relational databases: LegoDB (structure-mapping) [12], Monet [39] and XParent (model-mapping) [24]. All three approaches were modified to support spatial objects. The advantage of using relational databases to store GML documents is the availability of robust tools for data processing, such as disaster recovery, management services, concurrency control, and query optimizers. LegoDB works well for both queries involving large numbers of attributes and documents having large amounts of data. If the GML application schema is external to the relational database, there are considerable advantages to an XML DB. Otherwise, its advantages are greatly reduced. Table 2 summarizes the various approaches that can be used for storing GML data.

| | **Models** | **Example Implementations** |
|---|---|---|
| Approaches to storing XML documents | - XML DBMSs designed specifically to store XML documents | - Lore, Rufus, Strudel, Xhive, Tamino |
| | - Relational Model: Represents information as relations (tables) and queries are transformed to SQL. | - Oracle, XRel |
| | - Object Oriented Model: Represents information as objects and their attributes | - $O_2$, Object Store |
| Approaches to mapping XML documents to database schema | - Structure mapping | - LegoDB |
| | - Model Mapping | - Monet, XParent |
| Hybrid | - Combination RDBMS / XML DB | - Oracle XDB |
| | - Object-relational | - JAXB |

**Table 2: GML/XML data models.**

## 2.3. Differences Between GML versions 1, 2, and 3[2]

Currently, the OGC is in version 3.1.1 of GML. Although GML 1 lacked in functionality, it served as the first evolutionary step towards the better approach that is available today. Still widely used, GML version 2 provides facilities to handle simple features, such as linear geometries restricted to one or two dimensions. In spite of this limitation, it allows developers to model the real world using the features present in its specification. A city, for example, can be modeled as a collection of features described by type and value pairs. This organization made it simple to specify “Main Street” as a feature with property = “name” and value = “string.” Developers can work with as many features as necessary in order to make their application complete.

GML 3 incorporates more intricate structures than either of the previous versions, including support for complex geometries, spatial and temporal reference systems, topology, units of measurement, metadata, gridded data, and default styles for feature and coverage. This more granular hierarchy allows developers to select the schemas or schema components that are most appropriate for their work and focus on that subset. GML 3 contains a tool  that creates a tailored schema, using only the required components from the GML core schemas. It includes support for complex 3D geometries, 2D topology, temporal properties, and dynamic features. The *TimeReferenceSystem* class, for instance, allows users to select map visualization according to a historical period of interest. In this manner, a map of Berlin can be displayed with either its pre-WWII configuration or today’s layout. Table 3 summarizes the general differences among the three versions:

| | **Advantages** | **Drawbacks** |
|---|---|---|
| GML 1 | - Uses both XML DTDs and RDF<br>- Simple model | - No type inheritance depending on the profile chosen<br>- Lack of underlying semantics<br>- Little support for namespaces in DTD version |
| GML 2 | - Based on XML Schema<br>- Availability of existing supporting tools<br>- Better support for namespaces | - Support for only Simple Features<br>- No 3D constructs |
| GML 3 | - Complex geometries<br>- Topology<br>- Spatial and temporal SRS<br>- Better support for visualization<br>- 3D support | - Implementation complexity |

**Table 3: Differences in GML versions.**

[2] Denotes the entire family of releases 1.x, 2.x, and 3.x.

It is important to note that GML 3 remains compatible with the previous versions, though minor modifications may be necessary when porting applications between versions. Certain new constructs have been introduced and old ones have been deprecated. The OGC recommends that deprecated components not be use in new applications, since they may not be supported in future versions. *GML Envelope,* for example, provides identical model components as the older *GML Box*, which may remain in existing applications. GML schemas in version 3.x have been expanded significantly from previous versions. In order to accommodate more extensive functionality, GML 3 is eight times as large as GML 2. Application developers, however, have the flexibility to use only the basic required definitions along with the optional ones implemented by their applications.

## 3. **XML Parser and Query Language**

A GML document is fully readable by XML parsers, and can be retrieved by standard XML Queries. Certain GML parsers are already commercially available (e.g., Galdos Systems [1], Geotools.org [3]). Application designers must consider carefully which of the existing XML tools will serve their purpose with the maximum efficiency possible. This section discusses some of these considerations.

### 3.1. XML Parser

The W3C XML schema definition language has been used to define the contents of GML. A parser reads a GML document, validates it against the schema, and creates a representation of the document. GML does not need a special parser. In fact, XML parsers can be used for parsing GML files since GML is based on XML specifications [32]. Some available XML parsers are Xerces2 [10], XSV [43] and MSXML4.0 [4]. A software application should be able to understand the meaning of each element in the GML dataset, whether the element refers to a feature, a property of a feature, or a feature collection. Validation may not always be needed, but ideally this functionality should be available. The software uses a GML or XML parser to validate the data so that it conforms to the GML schema, and it should understand how the data has been defined in GML according to the specification and the application schema. This knowledge helps the application correctly interpret data. Large datasets often make data processing a challenging task. Any XML parser should be able to read the GML file character by character and then represent the data in a meaningful manner. This is likely to slow the performance of GML for storing and retrieving the GML documents.

In addition to XML Pull Parsers, there are two standard APIs that are currently used by software applications to parse GML documents: the Document Object Model (DOM) and Simple API for XML (SAX). Table 4 summarizes the features currently provided by DOM and SAX. The choice of a DOM or SAX parser for GML documents depends on the resource usage and efficiency. Many parsers can produce both SAX and DOM output. DOM builds a tree structure as it processes the data, which tends to require a large amount of memory in the case of spatial databases. In contrast, a SAX parser traverses the document sequentially, treating the document as a data stream. This tends to consume fewer resources and hence can be used for larger datasets. However, the SAX parser does not support random access of data, and thus may prove inefficient in the case of large spatial datasets. Various studies have compared the performance of these parsers for GML [41, 47].

| | **DOM** | **SAX** |
|---|---|---|
| Basic difference | Presents documents as a tree structure in memory | Presents document as a serialized event stream |
| Memory required | Relatively high | Significantly less, especially for larger documents |
| Queries | Better for joins | Better for point and range queries |
| Suitable for | Small documents | Large documents |
| Developed by | World Wide Web Consortium | Informal group of participants of the XML-DEV mailing list |
| Advantage for GML | Random access to data | Handles events that are less taxing on memory resources |
| Disadvantage for GML | In-memory processing makes handling large data sets prohibitive | API implementation complexity |

**Table 4: Comparison between DOM and SAX parsers.**

While SAX tends to be more efficient for point and range queries, DOM has better performance with join operations. DOM parsers can be less desirable for large GML documents because of the substantial memory usage. SAX parsers, on the other hand, can be inefficient in cases where a query involves a large number of attributes, though the combination and types of attributes can make a difference. Therefore, a parser ideally should combine the advantages of both DOM and SAX. The Galdos GMLSDK is an example that combines features from both approaches [1].

## 3.2. GML Query Languages

Even the well-known query approaches that work well with XML files do not always give acceptable results when applied to GML documents that contain a combination of numeric, alphanumeric and spatial data [17]. A GML query language must be flexible enough to support querying and retrieving such data. A candidate recommendation of XML Query (XQuery), a query language for XML, has been published by the World Wide Web Consortium (W3C) [15]. The goal of this recommendation is to seek developer feedback on a data model for XML documents, a set of query operators, and a query language based on these query operators, before a final recommendation is released.

Many query languages have been proposed for querying GML documents [16, 44]. Although GML may utilize the readily-available query languages developed for XML, these languages must be extended with spatial operators if they are to be used for GML. A specification of query language for GML based on extending the concept of XML-QL [39] was proposed by Corcoles and Gonzalez [16]. The authors provide a comparison of query languages currently available for XML, namely XQL (XML Query Language), XML-QL, Quilt, XQuery, and Lorel [13].

All query languages are based on an underlying data model that abstracts away from the physical representation of the data. The objects represented in GML are often more complex than those typically encoded in XML, since geographic objects have both spatial and non-spatial attributes. The data model for a GML query language therefore has to reflect this complexity, and the queries must follow suit. Table 5 summarizes some of the pros and cons of using XML versus GML query languages for GML documents:

| | **Advantages** | **Drawbacks** |
|---|---|---|
| XML Query Languages | - Full industry support<br>- Less rigid hierarchy than GML query languages<br>- Available in out-of-the-box RDBMS | - Must be adapted to include spatial capabilities<br>- Supports alphanumeric types only<br>- May not understand GML fully |
| GML Query Languages | - Standard for geospatial data<br>- Spatial constructs and joins<br>- Existing extensions to XQuery<br>- Ability to link related features<br>- Efficient filtering of desired features | - Still in research stages<br>- Dependent on a spatial data model<br>- Less efficient with non-spatial DBs (i.e., RDBMS) |

**Table 5: Comparison between XML and GML query languages.**

The queries for GML data can be either spatial or non-spatial. Any query language proposed for GML should therefore provide support for a mix of both. Non-spatial queries are similar to XML queries, since XML queries and documents involve alphanumeric data only. The OGC Filter Encoding Implementation Specification provides a common standard that queries can leverage to limit the amount of resulting data in a given transaction [50]. XML query language models can be extended to include the spatial query attributes of GML. This takes advantage of the existing XML query processing capabilities and at the same time provides the additional capabilities required for GML data processing. GML queries differ from XML queries as they tend to involve larger joins over large datasets. In addition, storing, indexing, and querying spatial data requires more abundant resources than those needed for relatively simple alphanumeric data [38].

XQuery has been designed to meet the requirements of an XML query language, as identified by the W3C XML query working group [7]. Vatsavai extended XQuery as a base for a GML query language due to its more robust functionality than other solutions [44]. It can cope with complex queries involving different types of joins and serves as the current standard. XQuery also allows extension functions that can include spatial operations such as *intersects*. Several other approaches for developing query languages have been proposed [11, 16, 44]. However, an important consideration when developing such languages is to decide whether to extend an already existing XML query language or to develop a new query language for GML. It is not clear that creating a completely new language would be either efficient or successful.

Another research area in the field of query processing is how to index GML data. Indexes contain data storage information that can be used to speed up searches [40]. As mentioned earlier, GML documents can either be stored as is or the data can be stored in a database and converted to GML when required. Existing spatial indexing techniques can be used for storing GML data in databases. Well-established approaches, such as R-Trees, have a distributed structure that is suitable for fast retrieval of spatial data, though such an implementation may consume a great deal of computation power. Other variations, such as R* Trees and $R^+$ Trees, insert objects in distinct paths, making them exclusive to a node. Z-Order B-Trees provide access control to objects while avoiding writing conflicts, though this may impose a high performance cost. Depending on the nature of the spatial data, Quad Trees (which can be combined with R Trees), may or may not be as suitable for spatial data as the other approaches, though they still offer a fast search method. Table 6 lists some of the more common indexing approaches currently available:

| **Indexing Approach** | **Advantages** | **Drawbacks** |
|---|---|---|
| R-Trees | - Efficient spatial data manipulation<br>- Nested multidimensional structure<br>- Nodes map to disk pages for easy access<br>- Balanced tree structure | - Needs extra filtering to remove redundant objects<br>- Less efficient for non-interval-shaped objects<br>- More disk access and computation |
| R*Trees | - Minimizes region overlap<br>- Avoid multiple search paths<br>- Uses reinsertion to improve storage use | - Large CPU time needed for reinsertion<br>- Needs efficient node-splitting for tree balancing<br>- Less robust filtering |
| $R^+$ Trees | - No object overlap for faster searches<br>- Searches follow single paths | - Can disperse data in more than one page<br>- May cause object redundancy<br>- Requires partition of data space to avoid overflow |
| Z-Order B-Trees | - Reorganizes itself after small changes<br>- Robust concurrency control mechanism | - Performance cost on insertion and deletions<br>- Less efficient for geospatial applications |
| Quad-Trees | - Quick access and manipulation of objects<br>- Good for recursive image processing<br>- Fast searching | - Large space requirements<br>- Overhead for linked lists<br>- Less efficient for high-dimensional data |

**Table 6: Comparison of indexing techniques for spatial data.**

The OGC Filter specification can be leveraged by constraining data for efficient query execution, which can be greatly enhanced by implementing special indexing techniques such as the ones listed in Table 6. An XML indexing scheme has been proposed in [29]. There are two approaches that can be used to search XML documents: searching value and searching structure. Accordingly, the indexes for XML documents are also divided into two categories: path indices and value indices. Path indices are used for regular path expressions, while value indices are used for locating objects in the XML documents. Alternatively, XPath, as part of the Extensible Stylesheet Language Transformation (XSLT), can be used for both purposes. However, standard tools that use these indexing techniques have yet to be developed specifically for GML. To date, some research has been done on spatial indexing techniques based on various approaches. Spatial indexes generally assume a structure that can support spatial data. Common approaches such as R-Trees and Quad-Trees may therefore be helpful, but the ultimate decision depends on whether the GML data is being directly deposited into the storage system, or whether it is first parsed and then stored as text or in some other format.

## 4. GML Visualization

One of the main bottlenecks when processing and generating maps is the difficulty of handling large amounts of geospatial data. In many situations, the large volumes of vector, raster, and other data types to be processed, transmitted, and rendered are prohibitive due to low transmission speed, limited bandwidth, the wide array of data formats, and a lack of optimized data structures. In this section, we discuss techniques that can usefully be applied for the visualization of GML data.

## 4.1. Using SVG for GML Visualization

There are several approaches to generating mapping products in either web or local environments. For example, SVG is a common two-dimensional vector graphics standard. SVG is a XML product that is well suited for context-sensitive mapping based on the layers that the system developer wants to make available externally. Table 7 lists two approaches to the visualization of geospatial data. In the first, the GML file is converted into SVG format through a parser in conjunction with styling tools. Once the SVG file is available, it can be used by a visualization tool in rendering the map, as illustrated in Figure 1.

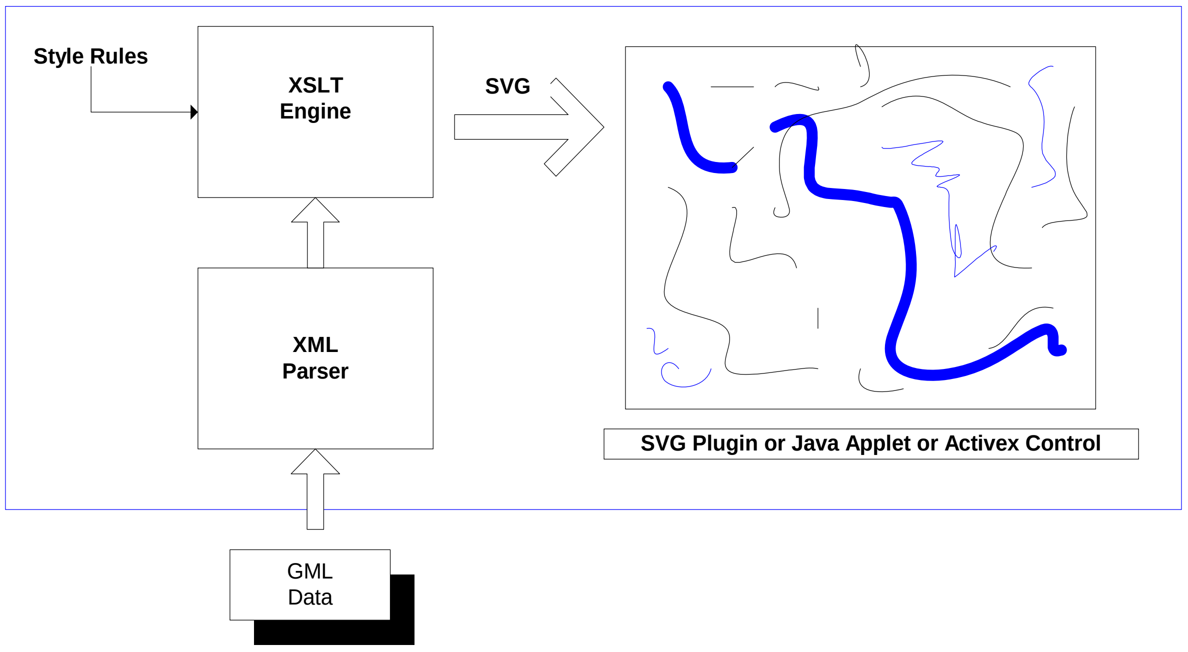

**Figure 1. SVG rendering process for GML data.**

The second alternative is to utilize a spatial database (e.g., Oracle Spatial or Arc SDE) and query the data based upon ad-hoc user requests to convert the SVG data into a format that the application can understand. This approach imposes a greater processing burden on the script itself, and less on the visualization tool.

| | **Advantages** | **Drawbacks** |
|---|---|---|
| GML data file =>SVG => Map | - More efficient conversion of large volumes of data<br>- Use of XSLT for easy styling<br>- Script can be customized for specific applications<br>- XML-only format easier to optimize | - Need for a conversion tool<br>- Reliance on an efficient visualization tool<br>- Less performance on large datasets<br>- Time to implement script |
| GML data file => Spatial Database => SVG => Map | - Handle large data sets<br>- More descriptive and flexible | - Cost of initial implementation<br>- More processing overhead than other approaches |

**Table 7. Visualization approaches for geospatial data.**

## 4.2. GML and XML Viewers

Data visualization is an important technique that helps in the understanding and analysis of complex data. Visualizing GML documents as maps is not in theory any different than displaying other graphics: the data is presented in a certain specific format and must be translated and displayed in order to represent real-world objects. Developers have the flexibility to choose their own visualization tools, whether based on GML, XML, or other proprietary formats. Some of these tools can be found off-the-shelf, some are developed in-house, and others are available as open-source code. There are many *gif*, *jpeg*, and *bmp* viewers, among others, that can be used for this purpose.

GML is not a visualization language. Although it is used to describe and store data content, it does not provide any information as to how the data is to be displayed. Graphical viewers enjoy wide industry support in both research and development, and can be used to handle GML documents. There are thus many available tools to choose from, each with its own distinct features. In addition, commercial as well as academic initiatives have explored a range of optimization techniques for XML files [49].

Visualizing geographical data is one of the primary goals of Geographic Information Systems (GIS). Graphical optimization can be achieved with standard XML viewers without any required changes to handle the newer constructs released in GML 3. No adapters are needed. Rule validation, for instance, can be enforced on GML constructs. Given the often high volumes of geographical data, performance can thus be improved and error handling enhanced, with more useful messages tailored for spatial data of the type that XML on its own cannot provide. Table 8 highlights some of the pros and cons of XML viewers.

| | **Advantages** | **Drawbacks** |
|---|---|---|
| XML Viewers | - Heavy industry/academic support<br>- Extensive R&D and testing<br>- Effective optimization techniques available<br>- Implemented by commercial RDBMS<br>- Error handling | - Varying product quality among different vendors<br>- Parsing performance on large data volumes<br>- Little support for spatial data |

**Table 8: XML Viewers.**

Before a map can be made from a GML document, the geo-spatial features must be extracted from the document and transformed to a suitable graphical representation with styling tools. The process of interpreting the GML data in symbols, such as line styles, area, and volume filling, is known as map styling. X3D (XML 3D), for example, is one technology that can be used for the presentation of GML data in a graphical format. The graphical representation is then rendered into a viewable image. An example that utilizes some of these technologies is AxioMap, an application using XML for Interactive on-line mapping [1]. Making a vector map on the web with GML data involves three steps: feature extraction, map styling, and graphic rendering [35]. To view an SVG or X3D data file, it is necessary to have a suitable graphical data viewer [26]. While some of the viewers have been built into Internet Explorer and Mozilla web browsers, several plug-ins are currently being developed for other applications.

An effective visual tool for the representation of geographic data should provide for the display and modification of maps [46]. Virtual Reality Modeling Language (VRML) describes 3D objects and virtual environments, allowing the representation of both static and dynamic data. GeoVRML is an extension to VRML that provides geoscientists with the ability to model 3-D geographic data that can be distributed over the web. Such data can be interactively visualized using a standard VRML97 browser configuration [37] and these features of GeoVRML facilitate widespread use of GML and geographic data by common users. GeoVRML provides several capabilities, such as precision, scalability, animation, and navigation, that have proven to be very important in representing GML data in a 3D world. Table 9 shows a comparison of various visualization techniques for GML documents. X3D [8], an open standards XML-enabled 3D file format, can also be a useful language in assisting GML data visualization. If used in conjunction with a rendering tool, it can translate GML directly into GIF, JPEG, Postscript, and other formats.

| | **GeoVRML** | **SVG** | **X3D** |
|---|---|---|---|
| **XML based** | No. Extends VRML to represent geographic data | Yes. XML-based vector graphics | Yes. Extends VRML97 using XML |
| **Representation** | Three-dimensional | Two-dimensional | Three-dimensional |

**Table 9: Comparison between various visualization techniques.**

## 5. GML in a Mobile Environment and GML Web Services

Two areas that have seen explosive growth in the past few years are the use of mobile devices and on-demand web services. GML has a high application potential in both of these areas.

### 5.1. GML in Mobile Devices

A mobile environment provides access to GIS data through wireless devices. It also gives users the ability to collect and transfer data using their mobile devices. Laptop computers, Personal Digital Assistants (PDA), such as palm pilots, pocket PC devices, and smart phones, are examples of the mobile devices in use today. Mobile GIS has been defined as a form of GIS that is specifically designed for users who are on the move, and is expected to be particularly useful for truck drivers, emergency response teams, and delivery personnel [28].

GIS is data-intensive, while mobile devices are data-sensitive. Often, mobile devices have only a limited capacity in terms of processing and storage. However, with recent improvements in wireless technology and reliable wireless communication, the ability to use GIS and GML 3 on mobile devices for easy access to spatial data is an area that is showing strong signs of growth. Commercially-available GIS services are already available for mobile devices. However, a range of enhancements can be made in this area, to address the constraints currently limiting handheld consumer devices (display resolution, display area, and memory), GPS guidance and orientation techniques, wireless network connectivity, interoperability among devices, data modeling (including searches in high-dimensional spaces, data retrieval and optimization), and real-time data management [23].

As mentioned previously, GML documents usually require a significant amount of memory due to their large size and a reasonable amount of bandwidth for efficient transmittal, both of which represent drawbacks that limit use in mobile devices. In an attempt to address these issues, a new language called cGML, or compact GML, has been developed specifically for mobile devices. The tags used in cGML are shorter in length than those in standard GML, which reduces the bandwidth and memory requirements by as much as 60%. This is not in fact a significant reduction, since compression can achieve better rates, but compression can be a computationally-intense task in a mobile environment. For instance, the FeatureCollection tag in GML is shortened to FtCl in cGML, and FeatureMember becomes FtMb [45]. Other important standards and proprietary systems that have been developed especially for use in mobile devices are C-HTML, XHTML, Web clipping, Handheld Device Markup language (HDML), Website Meta Language (WML), Mobile Execution Environment (MExE), and Wireless Application Protocol (WAP).

Another important issue currently limiting the use of GML on mobile devices is the display of geographic information. Geographic data and maps tend to be detail-oriented and may involve rich graphics and animations, most of which are likely to require server-side computing power. In addition, most mobile devices have a limited amount of screen space and provide only limited color resolution. A GML document therefore has to be transformed into a different format (e.g., SVG) to be displayed on the mobile device. An important consideration when designing wireless geographic systems is that wireless users are usually doing concurrent activities (e.g., driving a car, talking on the phone) and hence have a limited attention span for visual information display or for devices that require the use of hands such as a trackball [27]. Due to these constraints, the information presented to a mobile user has to be customized for the user's task, as well as for the device itself. The heterogeneous nature of mobile devices, in conjunction with the limitations imposed by the low-bandwidth and low-reliability wireless networks, has proven to be a challenge in such situations. GML could play a major role in providing such services in constrained environments. Table 10 lists some of the challenges that mobile devices impose on the usage GML.

| Hardware Constraint | Impacted Area |
|---|---|
| - Small screen size | - Graph rendering , image manipulation |
| - Low computing power | - Query performance , visualization of maps |
| - Limited bandwidth | - Retrieval of complex images , compression techniques |
| - Inadequate storage | - File size , data structure design , spatial data storage |
| - Small memory capacity | - Processing of large objects , cache design , spatial data extraction |

**Table 10: Challenges to the deployment of GML to mobile devices.**

The latest advances in mobile technology and computing are also likely to improve the integration of GML and wireless devices. The WAP forum developed the Wireless Application Protocol specification in order to provide a common platform for diverse wireless applications. Integrating GML into the protocol would be an important step towards the integration of GIS and mobile devices. Access to maps at any place during movement means that reliable Internet connections must be widely available. An important advance in this area has been the development of the GPRS (General Packet Radio Service) network by Telstra [1]. This network provides continuous connection to the Internet through a mobile handset, without any charges. A user is charged only for the time when he is actually using the Internet. This provides easy availability to the Internet without incurring high costs. Advances like GPRS provide reliable Internet access for mobile phones and pocket PCs, irrespective of location [9].

## 5.2. GML Web Services

GML has begun to play an important role in the development of GIS web services. Web services have been defined as modular self-describing solutions that clients can publish, locate, and dynamically invoke across the web [33]. The web services model provides users with only the services and data that they need. Users normally, and ideally, do not have to install, learn or pay for unused functionalities [10]. Such flexibility is important for providing easy access and introducing GIS systems to common users. However, such applicability and modularity also brings with it the problem of integrating the diverse systems and databases that will be using the services. GML provides the required support in such cases by providing a common medium of communication between the systems.

GIS web services have been grouped into three categories: data services, processing services, and registry or catalog services. Web Map Service (WMS), Web Coverage Services (WCS), and Web Feature Services (WFS) are examples of data services, as listed in Table 11. System designers often face various issues when implementing a WMS. Differences in versions can be troublesome, as they do not always conform in format, and may cause requests to fail. Resources are often referenced by non-standard namespaces, and once found, these resources are not always available with optimal retrieval performance.

| **Data Service** | **Functions** | **Applications** | **Issues** |
|---|---|---|---|
| Web Map Server | - Used for rendering maps or images. A web map server returns the image as an array of pixel values. | ArcView IMS, MapObject IMS, Galdos FreeStyler. | - Differences in constructs among versions.<br>- Non-standard namespaces to access resources.<br>- Need for performance enhancements. |
| Web Coverage Server | - A request to the server would return values that the client is interested in, namely, specific attributes of data such as temperature and rainfall. | IONIC RedSpider Web 3.1. | - Support beyond grid coverages.<br>- Support for coverage beyond pointing.<br>- Better retrieval of ranges, spatial subsets, and elevations. |
| Web Feature Server | - Provides an interface for spatial query requests including data manipulation requests and returning GML feature collections. | ArcExplorer (ESRI), IONIC Web feature server, Galdos CartaLinea, GeoMedia Webmap, CadCorp. | - Queries against multiple features.<br>- Support for querying heterogenous collections.<br>- Association between heterogeneous types. |

**Table 11. GIS web services (data services).**

A Web Feature server provides a set of geographic features that a client can use to perform data operations, such as getting or querying features based on both spatial and non-spatial constraints, creating new features, and deleting or updating a feature. The OGC Web Feature Service specification proposes interfaces for describing data operations on geographic features using HTTP as the distributed computing platform [34]. The specification requires that the Web Feature Service must use GML to express features within the interface. A standard query language or filters must be used to address geospatial queries to a WFS and obtain a collection of vector objects in a GML file in response [19]. Some of the issues that limit WFS are the little support for queries against multiple features and heterogeneous collections of features, as well as the unclear methods used to associate heterogeneous types. Nevertheless, WFS has become an efficient tool for GIS discovery. An example application that uses GML web services is the ArcExplorer by ESRI. ArcExplorer web

[20] is a open-source lightweight GIS data viewer that can be used directly from a web browser without downloading the application. Web Coverage Services is yet another type of service that returns data with its original semantics, so they can be interpreted further before being displayed. It also allows complex manipulation of spatial data, and the Coverages specification can  be enhanced for better support of ranges and elevations. Coverages are supported in GML by two schemas: coverage.xsd and grids.xsd. As specified by the OGC, applications implement coverages as either GML features or  objects containing the value of a property. Example 5 shows a code fragment where the feature is the town of Luzilandia. The coverage is represented by the drainage capacity for that town according to its rainfall amounts.

```
<loc:town  gml:id = “Luzilandia”>
        <loc:population>100000</loc:population>
        <loc:rainfallLevel>
            <loc:drainageCapacity> 25 </loc:drainageCapacity>
        </loc:rainfallLevel>
<loc:town>
```

**Example 5: Coverages.**

While WMS provides clients with the ability to render maps that are primarily static images, Web Coverage Services extends WMS  in several ways by providing not only the data, but also its descriptions, thus allowing the dynamic manipulation of features, and leaving the user to implement their own querying and interpretation of the information according to his or her application realm.

Other technologies that are used in the area of web services are Simple Object Access Protocol (SOAP), Universal Description Discovery and Integration (UDDI), and Web Service Definition Language (WSDL). SOAP is a XML-based protocol for information exchange among programs in a heterogeneous Internet environment using HTML and XML. UDDI is an XML-based registry that allows service providers or businesses to advertise their web services on the Internet, and WSDL is an optional XML-based language that is used by service providers to list their services in the registry. GML types (Application Schema) can be used in the types section of a WSDL document to define geographic objects that are part of the web service interface. GML can be a useful tool in the web services environment, since it is relatively easy to integrate it with other XML-based languages.

## 6. Discussion

GML is a powerful language that is highly suited to the storage, exchange, and modeling of geospatial data. It is non-proprietary, portable, and provides support for temporal data [22]. Features that change over time can be rendered as animated objects, which can provide a useful data-analysis tool. GML can encode most types of geographic information and can be sent to any device with an XML interface, greatly enhancing its usability in distributed environments [36]. The use of GML also helps integrate diverse XML-based geographic data with decision-making systems. For instance, in order to analyze the characteristics of a region, it may be useful to investigate its hydrography and vegetation, and then plot them in temporal layers to determine terrain change over time. Table 12 shows a summary of the current technologies in the field of GML, as well as the other topics that have been discussed in this paper:

| ***Area of GML/GIS*** | ***Languages, Specifications/ Standards*** |
|---|---|
| GML storage | Relational Model, Object Oriented Model |
| GML parsing | DOM, SAX |
| Query languages | XQuery, XML-QL, Lorel, Quilt |
| Visualization | GeoVRML, X3D, SVG, VML, XSL, CSS |
| Mobile applications | CGML, WAP integration with GML, XML hardware |
| Web Services | Web Mapping Service, Web Feature Server, OGC Web Server. |

**Table 12. Summary of current GML technologies.**

The efficient storage of GML documents is one of the problems that must first be resolved. Users have to decide if databases for XML should be chosen that use semi-structured data or if it would be more efficient to leverage existing relational databases. While both of these approaches have been shown to work, system designers must evaluate which would be most efficient for their specific application. Also, the sheer size of GML files poses a potential problem and new methods of reducing GML file size would certainly be useful. In addition, porting GML data into other formats raises several questions. Application designers must address whether GML on its own represents an effective storage approach and if adopting the OGC standards truly bring benefits in excess of the amount of generated overhead. RDBMS, ORDBMS or OODBMS each have their own pros and cons, and they serve different application niches, which must therefore be evaluated carefully.

As mentioned earlier, being XML-based gives GML some advantages as well as disadvantages. One advantage is that it can utilize many readily-available tools that have already been developed for XML. However, the associated disadvantage is that these tools are generalized, and are therefore not optimized for GML. Technologies such as DOM and SAX work well for XML documents, but their extension to GML can be problematic. The choice of parsers depends on efficient memory usage as well as the ability to retrieve random data, but XML parsers can be efficiently used with GML.

Query languages rely primarily on the efficiency of the data model and its operators. The design of a GML query language has to account for both spatial and non-spatial data. Queries can be significantly improved when used in conjunction with indexing techniques, such as R-Trees. While GML still relies on existing indexing techniques, there have been few initiatives to develop a native GML indexing scheme. One of the important features of GML is its ability to support the temporal characteristics of geographic data. This ability enables animated features to be used to represent the history of dynamic features. GML 3 provides support for a number of new features that were not supported by previous versions. For example, the latest version of GML has added new geometries and is backward compatible with earlier versions [2]. A particularly interesting new addition to GML 3 is its ability to represent geospatial phenomena in addition to simple 2D linear features, including non-linear 3D geometry, 2D topology and the relationship between features and geometric curves. However, the query languages that have been developed so far have not concentrated on the temporal aspects of the data.

Integrating GML with mobile device applications is a relatively new area. Mobile environments are inherently challenging due to their constrained nature. Adapting GML for low-bandwidth networks is a problem due to the detailed nature of geographic data. One must question what can be done to make GML data lighter for these devices in order to reduce storage requirements and maximize the usage of memory. At the same time, graphical rendering on small screens would be improved by a leaner version of GML. Finally, web services and GML will play an important role in providing GIS services to common users. WFS, WCS, and WMS all specify standard approaches for geospatial data manipulation. However, these technologies must still be enhanced with better functionalities to handle namespaces, coverage techniques, and queries on multiple objects. The development of GML as a standard raises several issues for which further research is needed, and these are listed below in Table 13.

| **Field** | **Research issues** |
|---|---|
| Storage | - Use of relational or object-oriented databases<br>- Implementation of native databases vs. databases specifically designed for GML<br>- Storage in GML files or other format<br>- Conversion of data to GML format<br>- Migration from proprietary format to open source GML<br>- Scalability to large datasets |
| Parsing | - Optimization of GML file processing with DOM and SAX<br>- Translation between different GML versions<br>- Comparison of GML vs. XML parsers<br>- Memory usage and support for random data access<br>- Conformance to and selection of suitable GML Schemas<br>- Enforcement of GML constructs, rule validation, message handling |
| Querying | - Handling of spatial and non-spatial operators<br>- Extension of query models for both spatial and non-spatial data<br>- Efficient retrieval of spatial data pieces, as well as full documents<br>- Indexing strategies for GML documents<br>- Implementation of GML-specific query languages<br>- Optimization of GML query languages |
| Visualization | - Visualization of 3D temporal data<br>- Conversion between different formats<br>- Application of SRS to different models<br>- Optimization of SVG for GML rendering<br>- Development of GML-specific visualization tools<br>- Extension of stylesheets for GML display |
| Mobile GML | - Compression of large GML documents<br>- Integration with other wireless technologies<br>- Spatial data structures for minimal storage<br>- Efficient visualization for small display screens<br>- Graphical rendering in low computing-power devices<br>- Adaptation of GML for constrained environments |
| GML web services | - Integration of GML with ebRim, UDDI and WSDL<br>- Expansion of namespaces<br>- Standardization of resource locators<br>- Response performance enhancements<br>- Efficient use of Coverage Features<br>- Support for multi-queries and type association |

**Table 13. GML research issues.**

## 7. Conclusion

Geospatial applications can benefit from GML's robust functionality and the technologies that enhance it. Several obstacles hindering GML usage, however, must be overcome so that maximum usability can be achieved. For this reason, we have described several areas of GML applications and some points of concern in this paper. The first consideration is the storage of GML documents, and the different database models that can be used to support GML data. The proper utilization of GML schemas is fundamental in the implementation of geospatial applications, as there are advantages and disadvantages to handling GML data in each of the available formats. Further, system designers must understand which parsers and query languages are suitable for GML, and which indexing strategies can be applied to GML documents. Visualization of GML documents can be used to leverage existing technologies, such as SVG, while making use of both standard XML and extended GML viewers. The increasing popularity of GML in mobile environments and the increasing availability of Web Services attest to the high potential of GML in these areas. Nevertheless, the undesirable limitations that accompany distributed and constrained environments must be addressed and mitigated. GML has the potential to emerge as a major form of geographic data processing, but before this potential can be realized, research must, at a minimum, overcome some of the issues presented here. Only then will GML become the de facto standard for geospatial applications.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their careful reading and in-depth suggestions.

## 8. References

[1] *Galdos Systems Inc. - http://www.galdosinc.com - Last Retrieved on May 19, 2006.*

[2] "Geography Markup Language (GML)," *Cover Pages:Online resource for markup language technologies. http://xml.coverpages.org/geographyML.html. Last Retrieved on May 19, 2006.*

[3] GeoTools.org - http://docs.codehaus.org/display/GEOTOOLS/Home - Last Retrieved on May 23, 2006.

[4] Microsoft Corporation - MSXML 4.0 SDK, *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk30/htm/xmmscxmloverview.asp*, Last Retrieved on May 19, 2006.

[5] Open Geospatial Consortium - OpenGIS® Filter Encoding Implementation Specification. Ref Number OGC 02-023r4 v3 , *http://portal.opengeospatial.org/files/?artifact_id=8340 - Last Retrieved May 19, 2006.*

[6] Oracle Corporation. Location-based Services for Oracle 9i, *http://www.oracle.com/technology/products/spatial/htdocs/data_sheet_9i/9iR2_spatial_ds.html - Last Retrieved on May 22, 2006.*

[7] World Wide Web Consortium - XQuery 1.0: An XML Query Language, *W3C Working Draft. http://www.w3.org/TR/xquery/. Last Retrieved on May 19, 2006.*

[8] XQuery 1.0: An XML Query Language, *W3C Working Draft. http://www.w3.org/TR/xquery/. Last Retrieved on May 19, 2006.*

[9] R. Ackland and S. Cox, "Markup Mapping," http://www.positionmag.com.au/GU/content/2001/GU47/gu47_feature/gu47_feature_2.html - Last Retrived on May 26, 2006. *GISUser - The Australasian Geographic Information Systems Applications Journal, Issue 47, September 2001.*

[10] Apache XML project, "The Apache Software Foundation - Xerces2 Java Parser," *http://xml.apache.org/xerces2-j/index.html - Last Retrieved on May 19, 2006.*

[11] D. Beech, A. Malhotra, and M. Rys, "A Formal Data Model and Algebra for XML," *University of California, Berkeley, CS 298-13: Digital Library Seminar. http://elib.cs.berkeley.edu/seminar/2000/20000207.pdf - Last Retrieved on May 19, 2006.*

[12] P. Bohannon, J. Freire, P. Roy, and J. Simeon, "From XML Schema to Relations: A Cost-based Approach to XML Storage," Proceedings of the 18th International Conference on Data Engineering, pp. 64-75, 2002.

[13] A. Bonifati and S. Ceri, "Comparative Analysis of Five XML Query Languages," *ACM SIGMOD Record*, vol. 29, pp. 68-79, 2000.

[14] B. Brodaric, S. Cox, J. Laxton, E. Boisvert, T. Duffy, B. Johnson, S. Richard, and B. Simons. "Standardizing Geologic Data Interchange: the CGI Datamodel Collaboration," GIS and Spatial Analysis - IAMG Annual Conference, 2005.

[15] D. Chamberlin, P. Fankhauser, M. Marchiori, and J. Robie, "XML Query (XQuery) requirements," *XML Query Working Group. http://www.w3.org/TR/xquery-requirements - Last Retrieved on May 19, 2006.*

[16] J. E. Corcoles and P. Gonzalez, "A Specification of a Spatial Query Language over GML," *Geographic Information Systems. Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems*, pp. 112-117, 2001.

[17] J. E. Corcoles and P. Gonzalez, "Analysis of Different Approaches for Storing GML Documents," *Geographic Information Systems. Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems*, pp. 11-16, 2002.

[18] S. Cox, P. Daisey, R. Lake, C. Portele, and A. Whiteside, "OpenGIS Geography Markup Language (GML 3.0) Implementation Specification," *OpenGIS Specifications. http://www.opengis.org/specs/?page=specs - Last Retrieved on May 19, 2006.*

[19] V. Dessard, "GML & Web Feature Server. The Baseline for Online Geoservices," in *Ionic Software Press Room.* *http://www.ionicsoft.com/pressroom/static/en/gml_wfs.pdf - Last Retrieved on May 22, 2006.*

[20] ESRI GIS and Mapping Software, ArcExplorer, *http://www.esri.com/software/arcexplorer/index.html - Last Retrieved on May 19, 2006.*

[21] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. "Catching the Boat with Strudel: Experiences with a Web-site Management System," *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pp. 414 - 425, 1998.

[22] Galdos Systems Inc, "Top 10 Benefits of Using GML," *Wireless Developer Network.* *http://www.wirelessdevnet.com/channels/lbs/features/top10gml/ - Last Retrieved on May 19, 2006.*, 2001.

[23] E. Gbei, J. Nafaa, I. Cosma, and M. Bernard, "Modeling the Scalable Vector Graphic ( SVG) Data for the Cartographic Generalization and the Multiple Representation on the Web," 2nd Annual Conference on Scalable Vector Graphics, 2003.

[24] H. Jiang, H. Lu, W. Wang, and J. X. Yu, "Path Materialization Revisited: An Efficient Storage Model for XML Data" *Australian Computer Science Communications*, vol. 24, pp. 85-94, 2002.

[25] L. Khan and Y. Rao, "Web Information Management: A Performance Evaluation of Storing XML Data in Relational Database Management Systems" *Proceedings of the Third International Workshop on Web Information and Data Management*, pp. 31-38, 2001.

[26] R. Lake, "Introduction to GML Geography Markup Language," *http://www.w3.org/Mobile/posdep/GMLIntroduction.html - Last Retrieved on May 19, 2006.*

[27] R. Lake, Galdos Systems Inc, "Location-Based Services & GML. Laying the Geo-spatial Web Foundations," 2001.

[28] N. S. T. Lee, "Single Line Street Network: The Foundation of Mobile GIS," Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 34-37, 1993.

[29] Q. Li and B. Moon, "Indexing and Querying XML Data for Regular Path Expressions," *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, pp. 361-370, 2001.

[30] A. Luniewski, P. Schwarz, K. Stamos, and J. Thomas, "Information Organization Using Rufus," *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 560-561, 1993.

[31] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widon, "Lore: A Database Management System for Semi-Structured Data," *ACM SIGMOD Record*, vol. 26, pp. 54-66, 1997.

[32] D. Murray and J. C. Chow, "An XML-Driven Data Translation Engine for GML 2," *Proceedings of the Urban and Regional Information Systems Association.*

*http://www.safe.com/company/media_archive/GML_User_Perspectives.pdf - Last Retrieved on September 28, 2005.*

[33] OpenGIS Consortium, "The OpenGIS Abstract Specification Topic 12: OpenGIS Service Architecture Version 4.3," *http://www.opengis.org/docs/02-112.pdf - Last Retrieved on May 19, 2006.*, 2002.

[34] OpenGIS Consortium, "Web Feature Service Implementation Specification," *http://www.opengis.org/docs/02-058.pdf - Last Retrieved on May 19, 2006.*, 2002.

[35] Z.-R. Peng and M.-H. Tsou, *Internet GIS Distributed Geographic Information Services for the Internet and Wireless Networks*: John Wiley & Sons, Inc., 2003.

[36] M. Prins, "Is GML only for Internet GIS?," *Directions Magazine. http://www.directionsmag.com/article.php?article_id=280 - Last Retrieved on May 19, 2006.*

[37] M. Reddy, L. Iverson, and Y. G. Leclerc, "Under the Hood of GeoVRML 1.0," *Proceedings of the Fifth Symposium on Virtual Reality Modeling Language (Web3D-VRML)*, pp. 23-28, 2000.

[38] P. Rigaux, M. Scholl, and A. Voisard, *Spatial Databases with Application to GIS*: Morgan Kaufmann Publishers, 2002.

[39] A. Schmidt, M. Kersten, M. Windhouwer, and F. Waas, "Efficient Relational Storage and Retrieval of XML Documents," Workshop on the World Wide Web and Databases (WebDB), 2000.

[40] S. Shekhar and S. Chawla, *Spatial Databases: A Tour*: Pearson Education, Inc., 2003.

[41] S. Shekhar, R. R. Vatsavai, N. Sahay, T. E. Burk, and S. Lime, "WMS and GML based Interoperable Web Mapping System," *Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems (ACM GIS)*, pp. 106-111, 2001.

[42] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, "XML Schema Part 1: Structures, W3C Recommendation," *http://www.w3.org/TR/xmlschema-1/ - Last Retrieved on May 19, 2006.*, 2001.

[43] H. S. Thompson and R. Tobin, "Current Status of XSV: Coverage, Known Bugs, etc," *http://www.ltg.ed.ac.uk/~ht/xsv-status.html - Last Retrieved on May 19, 2006.*, 2004.

[44] R. R. Vatsavai, "GML-QL: A Spatial Query Language Specification for GML," *Department of Computer Science and Engineering, University of Minnesota. http://www.cobblestoneconcepts.com/ucgis2summer2002/vatsavai/vatsavai.htm - Last Retrieved on May 19, 2006.*

[45] E. D. Vita, A. Piras, and S. Sanna, "Using Compact GML to Deploy Interactive Maps on Mobile Devices," *The 12th International World Wide Web Conference http://www2003.org/cdrom/papers/poster/p051/p51-devita.html - Last Retrieved on May 22, 2006*, 2003.

[46] A. Voisard, "Designing and Integrating User Interfaces of Geographic Database Applications," *Proceedings of the Workshop on Advanced Visual Interfaces*, pp. 133-142, 1994.

[47] W3C DOM WG, "Document Object Model FAQ," *W3C Architecture Domain.* *http://www.w3.org/DOM/faq.html - Last Retrieved on May 19, 2006.*

[48] M. Yoshikawa and T. Amagasa, "XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases," *ACM Transactions on Internet Technology*, vol. 1, pp. 110-141, 2001.

[49] X. Zhang, K. Dimitrova, L. Wang, M. A. Sayed, B. Murphy, B. Pielech, M. Mulchandani, and E. Rundensteiner, Rainbow: multi-XQuery Optimization Using Materialized XML Views,  Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 671, 2003.

## Appendix A

| Acronym | Description |
|---|---|
| 3D | Three-Dimensional |
| ADT | Abstract Data Type |
| API | Application Programming Interface |
| cGML | Compact GML |
| CPU | Central Processing Unit |
| DOM | Document Object Model |
| DTD | Document Type Descriptor |
| GIS | Geographical Information Systems |
| GML | Geography Markup Language |
| GPRS | General Packet Radio Service |
| GPS | Global Positioning System |
| HDML | Handheld Device Markup Language |
| HTML | Hypertext Markup Language |
| MExE | Mobile Execution Environment |
| OGC | Open Geospatial Consortium |
| OODBMS | Object-Oriented Database Management System |
| PDA | Personal Digital Assistant |
| RDBMS | Relational Database Management Systems |
| SAX | Simple API for XML |
| SOAP | Simple Object Access Protocol |
| SQL | Structured Query Language |
| SRS | Spatial Reference System |
| SVG | Scalable Vector Graphics |
| UDDI | Universal Description, Discovery, and Integration |
| VML | Vector Markup Language |
| VRML | Virtual Reality Modeling Language |
| W3C | World Wide Web Consortium |
| WAP | Wireless Application Protocol |
| WCS | Web Coverage Services |
| WFS | Web Feature Service |
| WML | Website Meta Language |
| WMS | Web Map Service |
| WSDL | Web Service Definition Language |
| X3D | XML 3D |
| XML | Extensible Markup Language |
| XSLT | Extensible Stylesheet Language Transformation |