

Web Integration of gOcad Using a 3d-Xml Application Server

Tobias Frank, Marcus Apel, Helmut Schaeben
Technische Universität Freiberg
3dxml@tobipascal.de
June 2003

Abstract

The background of this paper is situated in the area of integrating gOcad into a web based information system. Traditionally, an information system consists of the user application, the data storage and - according to the complexity of the information system - an application server. Current implementations of common application servers are designed for E-Business but not for complex calculations on multi dimensional geo spatial data.

In this paper we will show the implementation of a totally new 3d-XML Application Server (3d-XAppS) specially designed to interact with gOcad and an XML representation of the gOcad data model. The 3d-XAppS only calculates complex spatial requests, which cannot be solved by the means of the XML XQuery language using the gOcad data model. Furthermore this paper introduces an on the fly XML to GObj data converter designed for data interchange between the gOcad client and the database, respectively for any interface between an XML gOcad data model and gOcad Objects.

1 Introduction

Web services and information systems are common buzzwords in today's information technology. At Freiberg University we are integrating the gOcad geo modelling tool into a web based information system with enhanced GIS functionality. Core concepts of an information system are the simultaneous access of web clients to a central or distributed data storage and the calculation of complex requests. The data access is described by a query language. Relational database management systems use SQL (Structured Query Language) for example. But pure relational or object relational database management systems have many disadvantages for GIS back-ends. The difficulties arise with the structure of the modelled geological data. The best way to describe geo spatial data is by using a pure object oriented data model, in our case XML objects. XML (eXtensible Markup Language) is a Markup Language with the capability to describe custom data models. Another advantage of XML is that its text format is very comprehensible to humans. The query language to retrieve selected data from an XML data map is XQuery. Information systems often has to answer very complex requests, which cannot be solved by the means of the query language. Application servers calculate such complex requests. Common application servers use applications coded with interpreted programming languages or offer a vendor specific native programming interface. An interpreted code is far to slow for calculations on multidimensional geo spatial

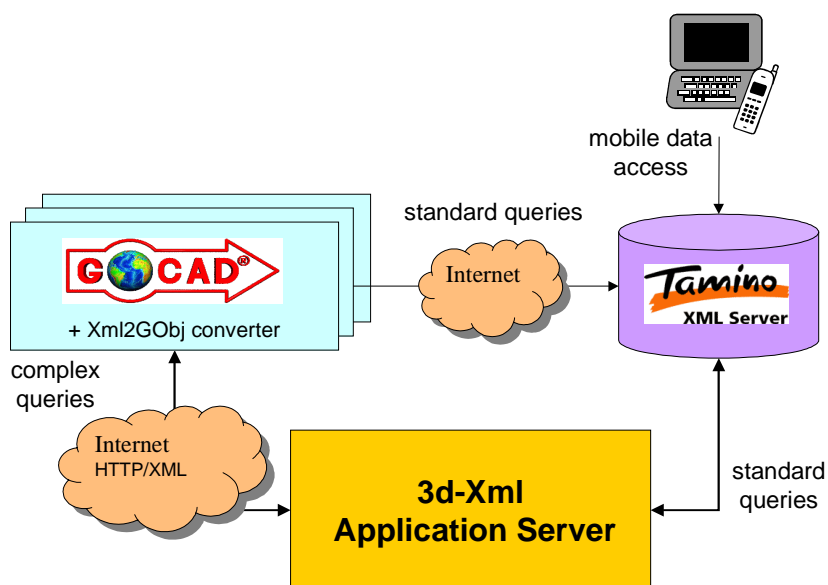


Figure 1: The 3d-XAppS is situated between a gOcad client and an XML database. Only complex requests are served by the 3d-XAppS. The communication between the gOcad system and the 3d-XAppS is based on HTTP and XML. The 3d-XAppS and the database management system are running on the same or on two different machines.

data, and using the application server's specific programming interface would increase the investment risk, because all development would depend on one product. Database management systems also offer programming interfaces, but one of our most important design goals is to be as vendor independent as possible. So we implemented a totally new 3d-Xml Application Server (abbreviated 3d-XAppS).

2 The 3d-XML Application Server

2.1 Web Integration, Production Environment

Figure 1 shows an overview of the whole information system. A *Tamino XML Server* is used for the data storage. At the moment, this database management system offers the most advanced XQuery implementation. The data-maps can be accessed by a geologist in the field using a lightweight database client, as well as by our modified gOcad system or the 3d-XAppS itself. Because the XML Server will always return XML data you need a converter to transform the XML representation of the gOcad data model to gOcad Objects (GObjs). This converter will be further described in section 3 of this paper. The gOcad client uses HTTP to communicate with the other components of the information system. For performance issues we recommend to run the 3d-XAppS on the same machine as the database management system. It is also possible to build a totally distributed system, but the transfer of huge amounts of XML data would reduce the overall performance tremendously. This is also one of the arguments for the usage of the 3d-XAppS against the integration of the application logic to gOcad. Many complex requests need a huge data volume for their calculation but only return a small result data map. An example for this

is the calculation of a collision between TSurfs which would only return the PLine of the collision. At the end you only need to transfer the PLine data over the internet and the calculation of the collision is done by the information system. This calculation is an example for a so called sub-sampling. The second group of calculations is called down-sampling.

Sub-sampling comprises set based, spatial GIS queries using topological and geometrical information for the calculation of containment, intersection or connectivity.

Down-sampling reduces the spatial resolution (mesh decimation). Down-sampling is used to retrieve macro structures from data with high resolution.

The 3d-XAppS is a large project and a complete explanation of its functionality would beat off this paper. Due to that we will concentrate on the main features and a description of its component based design.

2.2 Features

The list below shows the most significant features of the 3d-XAppS:

- The 3d-XAppS offers a high performant socket interface for communication with distributed components like gOcad and databases. The socket interface is working asynchronously and by that reserves systems resources.
- The 3d-XAppS offers a programming interface for 3d applications. These applications are based on lightweight software components, which can run in the 3d-XAppS's own process as well as in a 3d-XAppS independent context. The integration of distributed applications is possible, but not supported by this version.
- The 3d-XAppS offers a database driver programming interface for data exchange with a database management system. Again these drivers can run in the 3d-XAppS's or in an independent context.
- The 3d-XAppS can be run as a normal application or as a system service with the same code base and executable.
- The most important 3d-XAppS parameters can be configured by an XML file.
- There is no need for restarting the 3d-XAppS when a new 3d application or database driver is integrated to the system.

2.3 Design

The design of the 3d-XAppS is based on software components which interact with each other over well defined interfaces. Figure 2 shows the main components:

Socket-/Port Interface: The 3d-XAppS communicates with the gOcad system and the database over standardized internet protocols. Therefore the Socket Interface offers asynchronously implemented TCP/IP connection endpoints using the Winsock 2 [2] library.

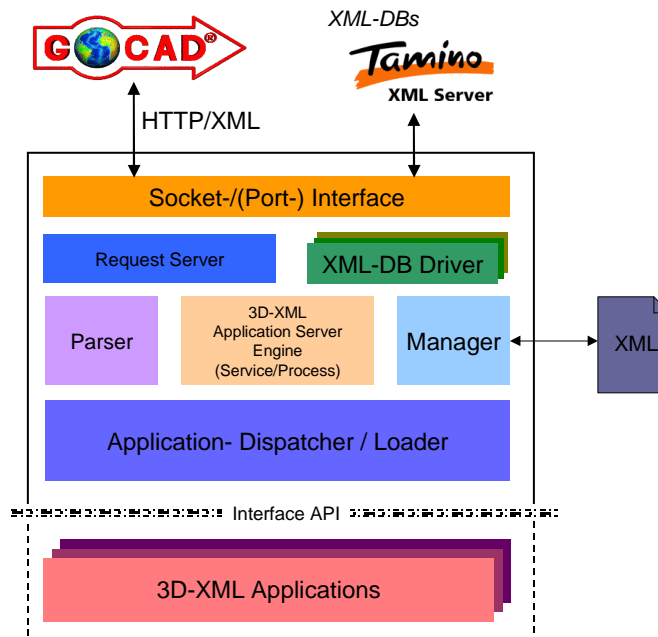


Figure 2: The 3d-XAppS consists of several components which interact with each other over well defined interfaces. The server's configuration is saved in an XML file. Any communication with the gOcad system or databases is based on TCP/IP and HTTP [1].

Request Server: The 3d-XAppS is receiving requests from clients and returns the results to them. The Request Server handles such I/O operations at the connection endpoints simultaneously. For the implementation of the Request Server we used parallel and asynchronous design patterns known for a high grade of scalability.

Parser: The 3d-XAppS uses XML technologies for the whole data exchange. You need an XML parser to process XML data. The parsers used by the 3d-XAppS are based on the Xerces-C++ [3] parser API.

XML-DB Driver: There is not one single database driver but you can use a lot of different database drivers for different database management systems. We implemented a programming interface for database drivers that enables the integration of different database drivers in a really easy way without restarting the server. For our scenario we developed a database driver for the *Tamino XML Server*. The option exists to code a database driver for the XQuery implementation of the *Oracle 9i* database or any other database management system.

Manager: The 3d-XAppS configuration is stored in an XML file. The Manager uses this file to extract the server settings used at server startup.

Server Engine: The Server Engine (codename *Ignition*) integrates the single components to the 3d-XAppS application. It includes the logic for both the standalone and the system service version of

the 3d-XAppS. Both versions are realized within one executable.

Application-Dispatcher / Loader: The Application Dispatcher loads the appropriate 3d application and dispatches the client request to the according application.

Interface API: The Interface API hides the complexity of the database drivers from the 3d application developer. It offers an easy to use database interface for retrieving, storing and updating XML objects. The Interface API also enables session based database transactions.

3D-XML Application: Again, there is not only one 3D-XML Application. Our idea was to develop a generic programming interface for the integration of 3d applications. New 3d applications can be used without the need of restarting the 3d-XAppS. Furthermore we developed a sample 3d application for collision detection between two surfaces.

3 XML to GObj converter

The purpose of the XML to gOcad Object (Xml2GObj) converter is to map the textual XML representation of geo spatial data onto C++ objects or data structures. For an optimal integration into gOcad the converter generates on the fly gOcad Objets.

3.1 Xml parsing models, DOM vs. SAX

There are two competing models for XML parsing. The Document Object Model (DOM), as well as the Simple API for XML Parsing (SAX) can validate an XML file against a DTD, respectively an XML Schema [4], [5].

Document Object Model (DOM): The DOM is a W3C standard that describes how to transform an XML file to a tree and offers methods to access and alter elements in this tree. The DOM parser generates a tree like illustrated in figure 3. The tree is stored in the memory and is an exact image of the former XML file. Copying the XML data into a tree representation and traversing the tree are very time and resource intensive. But the usage of the DOM API is very convenient.

Simple API for XML Parsing (SAX): The SAX is an event driven API, meaning that the occurrence of XML tags and element data is signaled by events. There are standardized callback routines which can handle the different kinds of events. You can override the callback routines to add additional logic for a custom parser. For our sample XML file the SAX parser generates the events illustrated in figure 4.

PCDATA values of an element are signaled by a `Character` event. When using the SAX parser, much more implementation has to be done instead of the DOM model, because you have to reimplement the event handler routines. Furthermore the events have no history mechanism. That means, you have to take care of the proper classification of an event into the document context. Parsing errors are also treated as events.

XML-FILE

```

<?xml version="1.0" encoding="UTF-8"?>
<gcm:GocadObject name="collision">
  <gcm:Model3D name="testModel">
    <gcm:TSurf name="tsurfA">
      <gcm:TFace>
        ...
      </gcm:TFace>
      <gcm:TFace>
        ...
      </gcm:TFace>
    </gcm:TSurf>
    <gcm:TSurf name="tsurfB">
      <gcm:TFace>
        ...
      </gcm:TFace>
      <gcm:TFace>
        ...
      </gcm:TFace>
    </gcm:TSurf>
  </gcm:Model3D>
</gcm:GocadObject>

```

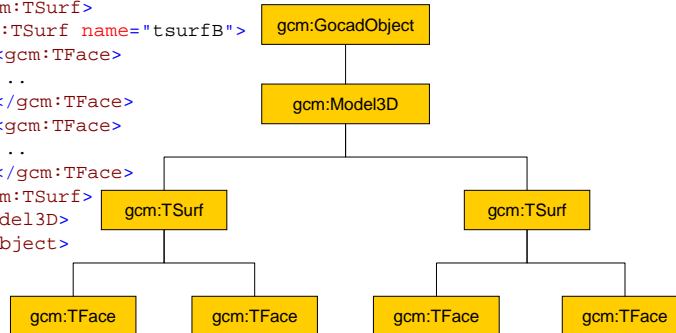
DOM-TREE

Figure 3: DOM representation of the sample XML file: The generation of the DOM tree in the memory is very resource intensive. The DOM API offers many operations to access and alter the elements of the DOM tree.

XML-FILE

```

<?xml version="1.0" encoding="UTF-8"?>
<gcm:GocadObject name="collision">
  <gcm:Model3D name="testModel">
    <gcm:TSurf name="tsurfA">
      <gcm:TFace>
        ... <!--PCDATA-->
      </gcm:TFace>
      <gcm:TFace>
        ... <!--PCDATA-->
      </gcm:TFace>
    </gcm:TSurf>
    <gcm:TSurf name="tsurfB">
      <gcm:TFace>
        ... <!--PCDATA-->
      </gcm:TFace>
      <gcm:TFace>
        ... <!--PCDATA-->
      </gcm:TFace>
    </gcm:TSurf>
  </gcm:Model3D>
</gcm:GocadObject>

```

SAX-EVENTS

```

=>StartDocument
=>StartElement
=>StartElement
=>StartElement
=>StartElement
=>Characters
=>EndElement
=>StartElement
=>Characters
=>EndElement
=>EndElement
=>StartElement
=>StartElement
=>Characters
=>EndElement
=>StartElement
=>Characters
=>EndElement
=>EndElement
=>EndElement
=>EndDocument

```

Figure 4: The SAX parser environment generates sequential events according to the structure of the XML file. The events are supposed to be handled by callback routines, the user can override this routines for customized functionality.

Table 1: Comparison between DOM and SAX

	DOM	SAX
Advantages	Easy to use API Permanent Access to XML data	High performant API
Disadvantages	Humble performance High Memory consumption	Implementation complexity No document context tracking

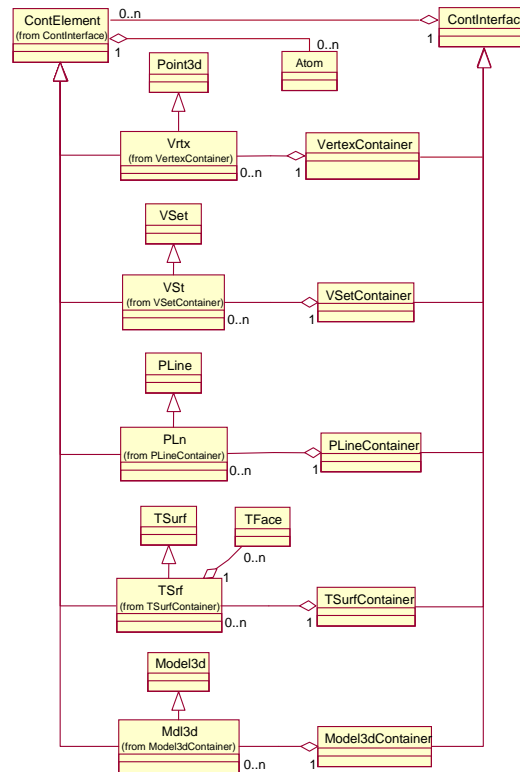


Figure 5: The geometrical elements of the data model are derived from an appropriate gOcad class and from the ContElement class. Geometrical elements that belong together can be aggregated in an according element container. Every element container inherits from the class ContInterface.

Discussion: After a brief introduction to two different parsing technologies let us draw a comparison between DOM and SAX, see also table 1. In respect to parsing large documents the disadvantages of the DOM API are not acceptable. Another argument for using the SAX parser is that the XML data will be parsed only once and will be directly converted into a C++ representation (gOcad Objects).

3.2 Xml2GObj Data Model

The current state of the Xml2GObj converter supports basic geometrical elements like points, lines and faces. So far there has been no support for solids. The basic elements are derived from the accord-

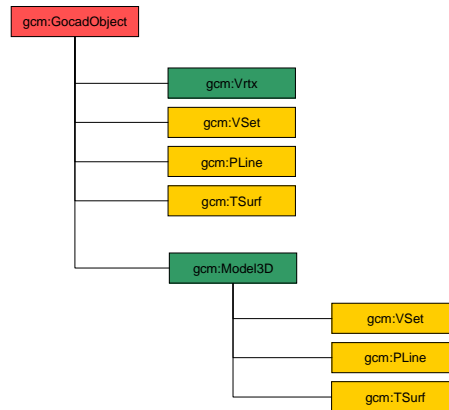


Figure 6: The first three hierarchies of the underlying XML gOcad data schema contain the macro structure of the geometry. Vertexes and Model3d only occur in the second hierarchy. Vertex sets, poly lines and surfaces can be globally defined in the second hierarchy or can be defined within a Model3d in the third hierarchy.

ing Gocad++ [6] classes and from the ContElement class. Elements which belong together are organized in an appropriate element container class. The container classes are derived from the class ContInterface. The ContInterface class realizes an associative container which stores the elements as key value tuples. There are different aspects for the usage of derived GObj (gOcad Object) classes. Because of the sequential work flow of the SAX parser, the geometrical objects need to store meta information like the accumulation of point primitives. But the most significant argument for custom GObj classes is the development of an advanced geological model, that is to be integrated into the overall concept in the near future.

When you analyze the underlying XML schema of the data map you can construct the tree shown in figure 6. This tree shows the first three hierarchies. The root element is `gcm:GocadModel`. The elements `gcm:Vrtx` and `gcm:Model3D` only occur in the second hierarchy. The elements for vertex sets, poly lines and surfaces can be child elements of `gcm:GocadObject` as well as child elements of `gcm:Model3D`. So they both occur in the second as well as in the third hierarchy. Elements of the same type which belong to the same parent element are organized in an element container. The class `Xml2GObj` aggregates the data model and the parsing logic. One `Xml2GObj` converts one data map. So the `Xml2GObj` class aggregates a `VrtxContainer`, a `VSetContainer`, a `PLineContainer`, a `TSurfContainer` and a `Model3dContainer`. The `gcm:Model3D` XML object can aggregate several point sets, poly lines or surfaces. So the according C++ class `Md13d` aggregates a `VSetContainer`, a `PLineContainer` and a `TSurfContainer`. This relationship is also illustrated in the UML model shown in figure 7. The nested class `XmlParser` realizes the parser logic.

3.3 On the Fly Conversion

The parser of the `Xml2GObj` converter is based on the SAX parsing model. In respect of large XML data maps it is necessary to avoid object copies and to use memory dynamically to improve performance and to preserve resources. All objects are generated dynamically on the heap and are destroyed when the `Xml2GObj` object will be destroyed. There is no difficulty in the handling the `gcm:Vrtx` XML

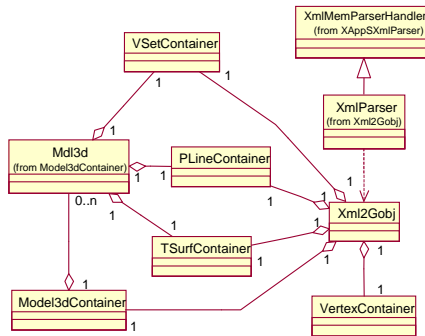


Figure 7: The `Xml2GObj` class contains both the data model as well as the parser functionality. Therefore it aggregates several container classes, which are storing geometrical elements. The nested class `XmlParser` realizes the parser functionality and is derived from the memory parser handler class `XmlMemParserHandler`.

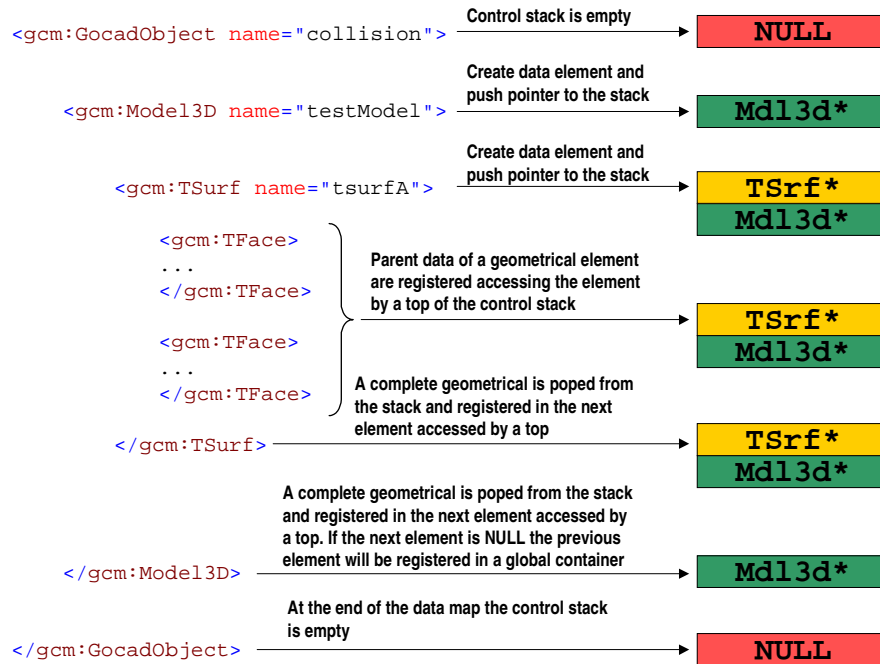


Figure 8: If a geometrical object (except `gcm:Vrtx`) occurs in the data map, an according C++ object is generated dynamically on the heap. All parent information of the object will be registered in this element. When an element is complete the pointer is pushed from the stack and the object itself will be registered in the next object in the stack or if the stack is empty the object will be registered in a global container.

object. Every `gcm:Vrtx` element belongs to the second hierarchy and is part of the `VrtxContainer` of the `Xml2GObj` object (see also figure 6). For the other geometrical objects introduced in figure 5, we

have used a control stack for identifying the proper context. This is essential because the XML objects `gcm:VSet`, `gcm:PLine` and `gcm:TSurf` can have global scope within a data map or can be nested in a `gcm:Model3D` object. If one of the geometrical objects, except `gcm:Vrtx`, occurs in the XML data the `StartElement` handler routine of the `XmlParser` class creates the according C++ object from the data model shown in figure 5. The object is created dynamically on the heap and a pointer of this object is pushed to the control stack. If an object is complete, its pointer will be popped from the stack and it will be registered in the top object of the stack. If the stack is empty, the object belongs to a global element container. This control stack realizes a context tracking mechanism and the geometrical objects will be registered to the according element containers. This tracking mechanism is also shown in figure 8.

4 Conclusion

In this paper we introduced technologies for the web integration of gOcad. At Freiberg University we developed a new application server designed for complex calculations on multidimensional geo spatial data. The kernel of the 3d-XAppS is not limited to 3 dimensions but can handle as many dimensions as described in the underlying data model. The component based design of the server enables an easy expansion with further database drivers and N-dimensional applications. A complete description of the design and implementation of the server and its programming interfaces can be found in [7].

To transform the XML representation of the gOcad data model an XML to gOcad Object converter is needed. The introduced object converter is a performant solution which generates gOcad Objects from its textual XML representation on the fly. The parsing model is based on the *Simple API for XML Parsing (SAX)*. Both the 3d-XAppS and the `Xml2GObj` converter enable to integrate gOcad to a web based information system which is based on standard communication protocols and today's data exchange formats especially XML.

References

- [1] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, third edition, 1996.
- [2] Jim Ohlund Anthony Jones. *Network Programming for Microsoft Windows*. Microsoft Press, Redmond, 1999.
- [3] The Apache Software Foundation. *Xerces-C++ Documentation*, 2001.
- [4] Eric van der Vlist. *Using W3C XML Schema*. O'Reilly XML.com, 2002.
- [5] Dare Obasanjo. *W3C XML Schema Design Patterns*. O'Reilly XML.com, 2002.
- [6] Association Scientifique pour la Geologie et ses Applications (ASGA), T-Surf. *GOCAD++ Developer's Guide*, 2002.
- [7] Tobias Frank. *Implementation of a 3d-Xml Application Server*. Technische Universität Freiberg, May 2003. Diploma Thesis, unpublished.