XML in Projects GNU Gama and 3DGI

Jan KOLAR, Denmark and Petr SOUCEK and Ales CEPEK, Czech Republic

Key words: GNU Gama, 3DGI, XML, XHTML, GIS.

SUMMARY

This paper presents our practical experiences with XML in geodetic and geographical applications. The main concepts and ideas of XML are introduced in an example of a simple web based information system, which exploits the XHTML language. Article further describes how XML is used in GNU Gama for structuring data for a geodetic network adjustment. In another application of XML, it is demonstrated how XML can be used for a unified description of data from leveling registration units. Finally, the use of XML for modeling 3D geographical features within the 3DGI project is presented and a relation of the project to GML and X3D is argued. The paper also aims at presenting our future plans with XML in our projects - namely the GNU Gama project, and the 3DGI project.

XML in Projects GNU Gama and 3DGI

Jan KOLAR, Denmark, Petr SOUCEK and Ales CEPEK, Czech Republic

1. INTRODUCTION

The XML technology is applicable and can be useful when one wants to store, present, read, or transfer data for further processing by other applications. The core idea of XML is to have the freedom of structuring and describing data, while, on a certain level, still using a standard technique. The level of being standard varies from case to case - depending on the context used.

The first goal of this article is to present different levels of being standard when using XML. This is shown on practical geodetical and geographical applications in order to clarify the use of XML in this domain. The presentation of these projects is the second objective of this article.

The article is organized into sections in the following manner. Section 2 serves an insight into XML from a practical point of view. The main advantages of using XML are shown on a small Web application. In Section 3, GNU Gama project and Project of complex processing of leveling observations are introduced. In Section 4 the 3DGI project and the possibilities of XML Schema, are presented. The whole article is summarized in Section 5, where also future directions are given.

2. XML AND XML APPLICATION

2.1 Introduction to XML

Extensible Markup Language [1], abbreviated XML, is a subset of SGML, the Standard Generalized Markup Language. XML is a standard and system independent way of representing data. Its main goal is to provide a common technology for exchanging data between systems over networks.

As an accepted and widely used standard, XML provides programmers with a complete set of techniques for handling data, i.e., XML documents, and with implemented tools, e.g., libraries or software components, for these techniques. In this Section, an introduction to concepts concerning XML standards and categories of XML documents is presented.

2.1.1 XML Compliancy and Standards

Often, when speaking about XML standards or XML applications, it is not clear what is exactly meant by these terms. One can ask; “what XML standards are, and what is an XML application?” To answer these questions is difficult, since they are ambiguous. This is due to the two-tier concept of XML technology. XML specifies a language for defining other languages, XML is a standardized specification possibly used for defining other XML-based

standard specifications. Further, XML specifies a well-formed document format that is required by all XML document type definitions. These XML definitions, which generate XML-based languages, can be called XML applications. However, any computer program that somehow exploits XML technology can in general be called an XML application.

It is very important to distinguish between the levels of the two-tier concept of XML, when speaking about standards and compliancy. The XML specification [1] is one standard on which many others are based. However, documents or software systems compliant with one of the XML-based standards are not always compliant with the other standards.

The applications introduced in this article are software systems implemented in one or more programming languages, e.g., C++ or PHP, which handle XML data. The software itself can also process or generate data of other formats. Nevertheless, concrete applications of XML, i.e., document type definitions, are introduced as well. Some of them are international standards, e.g., XHTML or GML, others are prototypes developed by us to suit the nature of problems being solved. Focus is put on numerous advantages served by XML.

2.1.2 XML Documents

The use of XML usually falls into two categories, data-centric and document-centric

- XML inherits from SGML property that separates presentation from content. If XML documents are to be published, the use of XSLT or CSS will lead to satisfactory results. In other words—an XML document can have various different representations, which can be controlled independently from the content. That is to say, XML documents should be data-centric. Data-centric documents are usually more structured and can be easily processed by computer programs. This approach is particularly suitable when transferring data to or from a database. Basically, most of the applications processing data-centric documents are related to or act as database systems.

- Nevertheless, XML also allows writing documents for human consumption, i.e., document-centric documents. Such documents could be emails, manuals or web pages. Such documents tend to be rather unstructured, but can easily be rendered on some output device, e.g. screen or printer. The structure of these documents might be irregular, and the order of each element is significant. Because of the irregular structure of document-centric documents, it is impractical to parse them. Instead, we usually store the whole documents into the database or file system.

The distinction between data-centric and document-centric documents is not always clear. Some technical reports might contain many kinds of fine-grained data, such as measurement records, statistical data or device specification. On the other hand, some surveying report may contain several human oriented notes or other loosely structured records. In these situations, the documents are both data and document-centric. The preference of the layout depends solely on the requirement of the applications.

2.2 Website Management Application with XHTML

Let us demonstrate some of the advantages of using XML in a small website management application named WebMachinek. This application was developed by Jan Kolar during his master thesis, initially meant for serving geographical information services on the Web; however it can be used for any Web site development.

The main features of the application are to manage the structure of the web pages as well as their content. The structure is hierarchical, which means that the user is conceptually allowed to divide the whole site into sections, sections into sub-sections and so forth. The content of each page is XML valid data according to an arbitrary XML document definition.

WebMachinek is written in PHP and Java Script. The output is an XHTML [11] document, i.e., a web page. Each returned XHTML document contains information about its position within the web site, e.g., a navigation bar, and the user-defined XML content. The content is always, if necessary, transformed to XHTML. WebMachinek is available, free of charge, at http://webmachinek.sourceforge.net

2.2.1 Exploiting XML Features

WebMachinek benefits from numerous advantages of XML, which are presented individually in the following text. The first advantage is that parsing and transforming XML data is supported in the programming language. Since PHP had built-in functions for transforming XML documents with XSLT processor, it was not necessary to implement such functionality again. Instead, just the mapping from one set of tags to the resulting set of tags is specified—the process of transformation is supported on the level of programming language. The same situation is regarding parsing XML documents in order to access relevant data by the application. Even though our application does not use an XML parser, the functionality is available.

The next good feature is the possibility of reusing existing XML document definitions, which suits the nature of our data well. If this is the case, the time and effort used for developing a data model is reduced considerably. WebMachinek exploits XHTML definition according to which the content of the pages is stored. Although XHTML is international standard, one can with the same efficiency exploit non-standardized XML definitions, which fit better to a given problem. Examples of the latter will be introduced later in the article.

The reuse of existing XML definitions helps us with the development of the data structure. However, the possibility of exploiting existing software or components is often even more beneficial. Due to the compliancy with XHTML, the WebMachinek does not need implementation of any rendering interface for presenting data. For this purpose, Web browsers, e.g., Netscape or IE can be used.

It should also be mentioned here that once you understand and are able to use XML technology, the same experience can be applied in diverse developments in the future. XML is not restricted to a particular programming language or specific kind of data. XML is a meta-standard applicable across computer systems.

XML is not the almighty technology for exchanging data. There are some disadvantages you have to be aware of. Firstly, one should take into consideration the requirements for space needed to store XML files and subsequently the amount of data, which has to be transferred in order to communicate with the files. Another possible inconvenience is difficulties with binary data that are not directly supported in XML files.

3. DATA CENTRIC DEFINITIONS GAMA-XML AND DNP

3.1 GNU Gama – Geodetic Network Adjustment

The open source project *GNU Gama* for adjustment of geodetic network was presented at FIG XXII International Congress [2]. More information can be found at [3]. The project was started in 1998 with XML used for description of structured adjustment input data.

A formal syntax of Gama XML input is provided by Document Type Definition (DTD). Apart from DTD, markup declarations can be described when using XML Schema; its possible advantages are discussed later in this article. DTD was used in Gama, partly because we had no experience with XML Schema, when the project started, but mainly because DTD is a well-established standard inherited from SGML.

When starting the Gama project, we did not face the problem of choice between DTD and XML Schema - the big question was which XML parser to use for processing XML data. From one point of view, we can divide the XML parser into two major categories: parsers implementing Document Object Model (DOM, the best representative is probably Xerces parser) and event driven parsers. For several reasons, we decided to use *expat* parser written by James Clark that falls into the latter category.

Compared with traditional data written in text or binary files, XML suffers from a substantial code bloat - apart from the raw data, a lot of XML tags need to be written and/or read. But is this a real problem? Let us take an example of a sparse matrix written by rows with column indexes and nonzero coefficients

```
600 803

5 1 3 2 5 4
0.01 -5.68573 -1 6.54948 -0.985854 -6.54948 0.985854
5 1 3 2 7 6
0.01 5.68573 -1 -5.83394 0.0218662 5.83394 -0.0218662
5 8 5 4 10 9
0.01 5.246 -1 8.21746 -0.609313 –8.21746 0.609313
5 8 5 4 3 2
0.01 0 -1 -6.54948 0.985854 6.54948 –0.985854
5 8 5 4 12 11
0.01 -6.08486 -1 -9.05865 1.55339 9.05865 –1.55339
5 13 10 9 15 14
0.01 7.15147 -1 6.7523 -1.16378 –6.7523 1.16378
5 13 10 9 5 4
0.01 -7.15147 -1 -8.21746 0.609313 8.21746 -0.609313

…… remaining text ignored ……
```

XML representation of the previous sparse matrix is

```
<?xml version="1.0" ?>
<!DOCTYPE gnu-gama-data SYSTEM "gnu-gama-data.dtd">
<gnu-gama-data>
<adj-input-data>
 <sparse-mat>
 <rows>803</rows> <cols>600</cols> <nonz>3401</nonz>
 <row> <nonz>5</nonz>
 <int>1</int><flt>-1</flt>
 <int>3</int><flt>6.54948</flt>
 <int>2</int><flt>-0.985854</flt>
 <int>5</int><flt>-6.54948</flt>
 <int>4</int><flt>0.985854</flt>
 </row>
 <row> <nonz>5</nonz>
 <int>1</int><flt>-1</flt>
 <int>3</int><flt>-5.83394</flt>
 <int>2</int><flt>0.0218662</flt>
 <int>7</int><flt>5.83394</flt>
 <int>6</int><flt>-0.0218662</flt>
 </row>
 <row> <nonz>5</nonz>
 <int>8</int><flt>-1</flt>
 <int>5</int><flt>8.21746</flt>
 <int>4</int><flt>-0.609313</flt>
 <int>10</int><flt>-8.21746</flt>
 <int>9</int><flt>0.609313</flt>
 </row>

 …… remaining text ignored ……
```

The following table is summarizing results of some sparse design matrices from geodetic network adjustments

size	*Obs*	*pars*	*res*	*xml*	*res.gz*	*xml.gz*
1386	63	22	3912	16671	1233	2017
2415	69	35	3974	18001	1197	2068
2880	96	30	6350	25609	1881	2878
8056	106	76	6817	29382	1946	3093
11610	129	90	7472	34096	2116	3415
11664	144	81	8692	36045	2809	4317
11750	250	47	18436	73589	3309	5248
481800	803	600	54682	220799	16271	24342

The columns *obs* and *pars* contain numbers of observations and unknown parameters, *pars* and *res* are sizes in bytes of sparse project equation file and corresponding XML representation; the last two columns are the sizes of these files compressed by GNU gzip. Clearly, for growing network size the ratio of compressed sparse project equations (*res.gz*) and compressed XML (*xml.gz*) is low. When the volume of transmitted data might become critical, we can always use compressed files for which the difference in sizes is not crucial. Furthermore, compared with raw data (*res*) the XML representations are almost self-documenting.

GNU Gama was formerly started as an educational project designed merely for adjusting local geodetic networks. This project branch is more or less closed, and this year we have started working on a new development branch aimed for adjusting a global coordinate system (with the adjustment model based on spheroid). For this new development branch new data structures are needed, and existing XML input format will have to be replaced by a more general one.

Exploiting XML in our project as just an input format did not prove adequate; all communication within the project should be available in XML (namely the adjustment results). This premise shows that a single DTD is needed for the Gama project. It would be very cumbersome to have various DTDs for various modules in one project.

Designing a general DTD (or XML Schema as discussed below) for the whole project is not a trivial task. The ideal solution would be if we could use a general XML application. Borland *data packet* (Borland C++ Builder), shown in the following example, seemed to be a good candidate, because it can be used for any data, described in its introductory metadata section. We did not use this model, mainly because it was not clear if it could be used in an open source project.

```
<?xml version="1.0" standalone="yes"?>

<DATAPACKET Version="2.0">

 <METADATA>
 <FIELDS>
 <FIELD attrname="point" fieldtype="string" WIDTH="20"/>
 <FIELD attrname="type" fieldtype="string" WIDTH="1"/>
 <FIELD attrname="y" fieldtype="string" WIDTH="15"/>
 <FIELD attrname="x" fieldtype="string" WIDTH="15"/>
 <FIELD attrname="v" fieldtype="string" WIDTH="15"/>
 </FIELDS>
 <PARAMS/>
 </METADATA>

 <ROWDATA>
 <ROW point="1" type="fixed" y="644498.590" x="1054980.484"/>
 <ROW point="2" type="adjusted"/>
 <ROW point="407" type="unknown"/>
 </ROWDATA>

</DATAPACKET>
```

New DTD for Gama is designed to describe a series of data object in one XML document (parser will return a list of data object to its calling application); as an example of this, please see the XML representation of a sparse matrix *(adj-input-data)* in this section. Apart from others, this concept was chosen because it enables easy implementation of a client-server model.

In the new development branch, XML is not limited to external communication but is used to define project parameters as well. For example supported ellipsoids are described in an XML file. From this file, tables are generated, which are used in the documentation and C++ classes for handling ellipsoid parameters (would it be possible to describe cartographic projections and/or datum transformations this way?). Multilingual versions are generated.

3.2 DNP – XML Definition for Leveling

In this article, we would like to present another example of an XML application, namely a unified XML format for describing leveling observations from various types of digital registration units. This format was named DNP by its author, Petr Soucek.

Different companies, producing digital registration units, use their own different proprietary data formats for leveling data. From the user's point of view, the absence of a uniform data format is clearly a drawback. Geodetic companies face problems, when using different hardware. Thus it is difficult to share data between incompatible leveling instruments. The situation described was a motivation for the effort to design a unified leveling data format DNP, based on XML, into which it would possible to convert data from the typical widely used leveling instruments (Leica Geosystems, Spectra Precision - Zeiss, Sokkia, Topcon).

Apart from the leveling observations, DNP enables storing information about the data such as name of the surveyor, weather conditions etc.; it should be stressed that in order to enable possible future enhancements, it is necessary to add information on the DNP version used. One of the advantages of this XML format is that it is an easily readable text format, it is clearly defined by the DTD, and it is platform independent. Two main components of the DNP format are:

- system headers (information on DTD, input file, creation time and date)
- description of the file and observations; this section is divided into subsection describing instrument, surveyors, weather, a subsection of observed data. Leveling lines are written by points; point data can contain all relevant information.

Let us briefly mention definition styles which enable transformation of XML documents into another document type (XML corresponding to another DTD, HTML, XFTML, RTF, WRML, plain text, …). Clearly this can be achieved by writing our own software, based on an XML parser, but in many cases using XSLT style language is much simpler. We are using XSLT languge in our project for conversions from DNP into HTML, the transformed document is rendered in a web browser as a classical leveling form.

3.2.1 Project of Complex Processing of Leveling Observations

The DNP format was designed by Petr Soucek as a basic XML data format used in the project of new complex technology for the processing of leveling observations developed in collaboration with the Land Surveying Office in Prague [4]. In this project, all leveling data from the registration units are converted into the common DNP format that is subsequently

used in adjustment and analysis. The software package developed contains the following modules

- digital transfer of observed leveling data into the computer,
- digitalization of the coordinates of leveling points from the map and transformation into projection plane (used only if coordinates are not known),
- organization of leveling lines and their conversion into the DNP format (and subsequent transformation into XHTML),
- adjustment and analysis of selected leveling lines and/or leveling networks with optional conversions to DBF format and XLS Microsoft Excel format and graphical outputs.

The software is written in C++ (using *expat* XML parser by James Clark) and Windows GUI built with C++ Borland Builder 5. All programs (in Czech version only) are available, free of charge, from
http://gama.fsv.cvut.cz/~soucek/archive/nivelace

4. 3DGI PROJECT

In the following section, a project within a Centre for 3D GeoInformation is introduced. The main focus of the project is to lay down a background for the development of systems for Geographical Information Science [7] (GISc). The result of this effort should be a framework, reusable for development of possibly interconnected geographical information systems dealing with 3D.

Focus is particularly on the representation of fundamental geographical features in 3D, which should be common for data in both global and detailed scale. This aim requires both interpretation and implementation of knowledge from the physical geodesy in terms of a database and information technologies. One of the core subjects is to provide a data structure, representing two main geographical 3D features---a geoid and a terrain surface. Moreover, the representation has to provide the possibility of gradual refinement, i.e., adding new data concurrently when it being available from new measurements. The representation is also required to keep track of the accuracy and the temporal properties, as well as their ability to co-exist with other objects that can be possibly related to the terrain surface and/or geoid. That is why it is highly required, at least during the development process, to use a flexible but still powerful means for both handling and structuring data. XML Schema [8] is powerful not only in defining XML structures, but also in providing data type capabilities to XML, adding a measure of object oriented programming support. Another good thing is that Schema supports integrity constraints, including multi-attribute primary and foreign keys or minimum and maximum cardinality constraints. These features are very useful for the database schema design and mapping between XML and DBMS. Next sections introduce shortly XML Schema and sketch its use in the 3DGI project.

4.1 Reasons for XML Schema

It is said that in a typical program, up to 60% of the code is spent checking the data. DTD is sufficient in structuring, but it is quite constrained in checking the data.

First of all, DTD does not even support some of the primitive types, such as boolean or integer, whereas XML Schema supports more than 44 data types. We could also define our data types in XML Schema.

DTD uses the syntax from SGML, which is different from XML. It is necessary to use two syntaxes in order to validate the XML documents. Schemas are written in XML. Because XML is extensible, Schemas also undertake this feature. We can reuse one Schema in another one or create our own data types derived from the standard types. It is possible for example to constraint values of an angle to be between 0 and 400, when grad units are used.

Furthermore, DTD has no support for inheritance. However, Schema provides an analogy to this mechanism of OO-programming. It is possible to define a common base type for an element (object), and then extend the type to a larger and more specific type. Schema provides us with a way to extend and restrict an existing type in order to accomplish inheritance.

Additionally, Schema provides a mechanism of referential constraints, which are used in the database system. The use of key and keyref simulates the concept of primary and foreign keys in the database systems. Schema also provides better content modeling mechanism. It has the ability to constrain the order and number of child elements, and is able to define a minimum and/or maximum number of consecutive instances of an element.

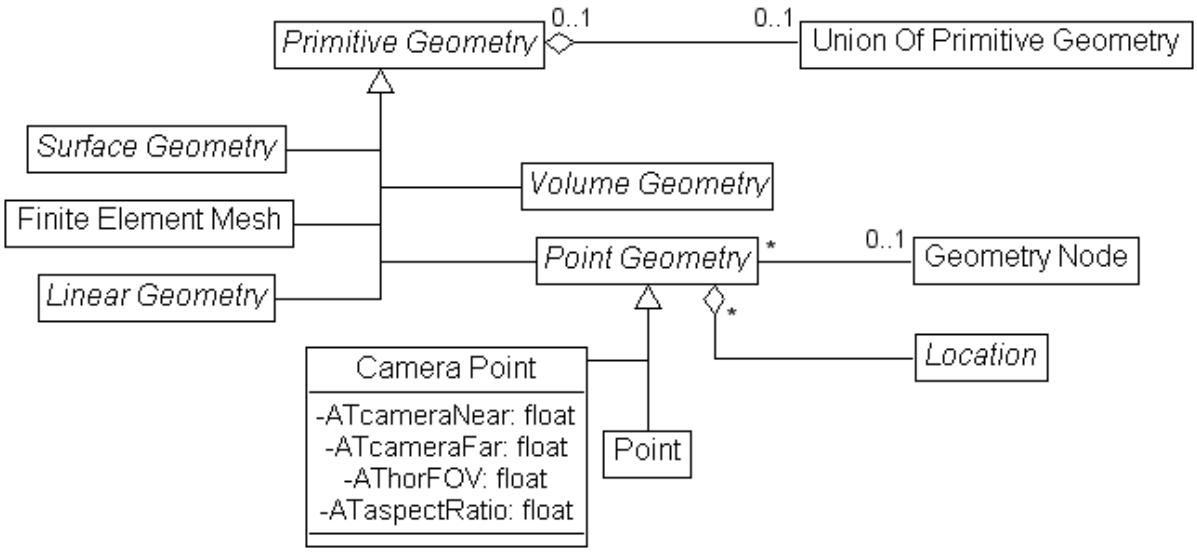
Finally, in Schema every name is associated with a namespaces. That way we can have the same name for different elements without causing a conflict. For example, we can define “name” for a geodetic point and “name” for a geodetic network.

Nevertheless, it has to be mentioned that DTD is widely used and for many cases it is sufficient. There is no need to change working DTD to XML Schema unless the reasons mentioned above bring a substantial improvement. This is mostly concerning DTDs developed to work with a specific application. However, for developing a common XML document types Schemas are a much more powerful mechanism, particularly when dealing with elaboration of a complex data model. A simpler DTD can be derived from Schemas afterwards, if necessary.

4.2 XML Schema in 3DGI

Since data representation is one of the major problems in 3DGI, it is necessary to elaborate a conceptual and logical model. A logical model describes the physical data structures in an abstract way, which is often expressed graphically through ER diagrams or UML diagrams

[5]. In the following figure, an example of logical level UML class diagram from 3DGI project is depicted.



For reasons mentioned in Section 4.1 it is convenient to have a direct link from the logical level to a physical model defined in XML. As introduced in the previous section, XML Schema has numerous powerful features, which are sufficient for creating XML definitions for most conceptual models.

Further in this Section, the mapping from UML class diagram to XML Schema developed in 3DGI is described. In order to see resulting XML Schema for the diagram from the figure above, look up in Appendix A. For XML Schema terminology is used monospaced font. 3DGI logical level class diagram is mapped on XML Schema as follows:

- UML Class, i.e., a rectangle on the figure, is mapped to `element` definition in Schema.

- UML Class Attribute is mapped to *element* with *simpleType* or *complexType*. The elements that correspond to UML Class Attribute can be distinguished through naming convention, e.g., names start with string “AT” in our case.

- UML Abstract Class, i.e., a rectangle with the name in italics, is mapped to an *abstract complexType* declaration. If UML Abstract Class is associated or aggregated with another Class, than it is also mapped to *abstract element*.

- An inheritance relationship, i.e., a line starting with a triangle, is mapped as *extension/restriction* of *complexType* of parent UML Class. If a child UML Class is also mapped to *element* and parent UML Class is mapped to *abstract element* than *substitutionGroup* is defined in the child *element*.

- An association relationship, i.e., a straight line, and an aggregation, i.e., a line starting with diamond, are mapped as follows. Any associated or aggregated UML Class is mapped to *element* in *sequence* declaration within the related Class.

- A multiplicity, i.e., cardinality information attached to the ends of lines, is mapped through *minOccurence* and *maxOccurence* constraints.

The solution does not cover UML Class diagrams completely. Multiplicity constraints of parent Classes, i.e., from child perspective, for aggregated, inherited and one-way associated Classes are not preserved. For example, it is not covered in our Schema that one instance of 'Location' can be related with zero-to-infinity instances of 'Point Geometry'. It is also impossible to distinguish between one way relationship and aggregation from the resulting XML Schema. However such information is it not kept in implementation by programming languages neither.

4.3 The use of Standardized XML Definitions in 3DGI

The first XML-based standard dealing with geographical features was the Geographical Markup Language (GML) [10]. Current GML version 2 (version 3 is being released at the time of writing this article) is an XML Schema definition, which implements OpenGIS Concortium's (OGC) "Simple Feature Geometry." Even though coordinates can be specified in three dimensions, GML provides no direct support for three dimensional geometry constructs. Apart from primitives, e.g., point, line and polygon, geometries can also be homogeneous collections, e.g., multi-point, multi-line string and multi-polygon, or heterogeneous geometry collections.

Geometric properties are restricted to geometries for which coordinates are defined in two dimensions. The delineation of a curve is also subject to linear interpolation. For these reasons, GML is not sufficient for needs of 3DGI. The representation of 3D surfaces, e.g. terrain or complex roofs, provides a particular problem, for which grid data structures are suitable. However, there is no support for grids in GML. Furthermore, the GML does not deal with visualization of data, but it is possible to use transformation to SVG, which is an XML data type definition for 2D vector graphics.

Another XML-based standard, which is still being developed, is X3D [9]. It is the successor of VRML, and its scope is not bounded to language definition only. X3D defines the whole API for systems for rendering interactive 3D content and multimedia. In other words, the X3D definition of XML document type is only part of a broader effort on system standardization within the domain of 3D graphics. X3D integrates support for geographical systems. It provides the ability to embed geospatial coordinates, to support high-precision geospatial modeling, and to handle large multi-resolution terrain databases. These are common concepts that can be undertaken also in 3DGI project. However, X3D does not cover more specific problems of representing the real environment including browsing a huge model, e.g., a model of the globe, gradual update of data elements, classification of environmental features, e.g., road, town, restaurant, or automated multi-representation, e.g., algorithms for generating diverse level of detail.

Even though X3D is still in a state of evolving a proposal we can expect that it will provide a new XML definition for visualization of 3D graphics with rich experience from VRML. This fact would lead to a valuable exploitation of future X3D viewers in 3DGI project.

5. CONCLUSIONS AND FUTURE WORK

This article presents a practical insight into exploitation of XML technology in domains of geography and geodesy. Three projects dealing with geodata are introduced from perspective of dealing with XML data. The scope of GNU Gama is the adjustment of geodetic networks with XML input format defined in DTD. The project of complex processing of leveling observations is based on DNP - DTD definition describing leveling observations from various types of digital registration units in a uniform way. The 3DGI project exploits XML for development of representation of 3D geographical features as a flexible technology for traversing from a logical to a physical data model. Argued are also the possibilities of XML Schema for definition of new document types, and the exploitation of standard XML definitions namely XHTML, GML and X3D.

The future development of GNU Gama will focus on the adjustment in global 3D coordinate system. This will end up with a new XML definition, which should also contain a description of an output format. The project of complex processing of leveling observations is already dealing with implementation of DNP in practice. The 3DGI project continues developing XML Schema representation of 3D geographical features, which is suitable for navigation through huge 3D models.

Despite specific goals of introduced projects, a substantial benefit from XML is present in each of them namely for processing and communicating data. It proves that XML is convenient in geodetic and geographical applications.

REFERENCES

Extensible Markup Language (XML) 1.0, W3C Recommendation 10-Feb-98, http://www.w3.org/TR/REC-xml

Cepek, A. – Pytel, J.: Free Software – an Inspiration for Virtual Academy, TS2.1: Virtual Academy–Case Studies and Experiences, FIG XXII International Congress, Washington, D.C. USA, April 19-26 2002

GNU Gama, http://www.gnu.org/software/gama/

Department of Leveling and Gravimetry, Land Surveying Office, Czech Office for Surveying, Mapping and Cadastre

SEDRIS: Data Representation Model Notation, http://www.sedris.org/dm_notn.htm

Je-Luen Tzeng: Transferring Data Between XML Documents and PostgreSQL DBMS, Master Thesis at Indiana University USA, May 2002.

Jonathan Raper: Multidimensional Geographic Information Science: Taylor and Francis, 11 New Fetter Lane, London EC4P 4EE, UK, 2000.

David C. Fallside (Editor): XML Schema Part 0: Primer, W3C Recommendation May 2001, http://www.w3.org/TR/xmlschema-0

X3D, Web 3D Consortium, February 2003,http://www.web3d.org/fs_specifications.htm
Geography Markup Language (GML) 2.0, OpenGIS® Implementation Specification, 20 February 2001, http://opengis.net/gml/01-029/GML2.html
XHTML™ 1.0 The Extensible HyperText Markup Language, W3C Recommendation 26 January 2000, revised 1 August 2002, http://www.w3.org/TR/xhtml1

CONTACTS

Jan Kolar
Centre for 3D GeoInformation
Aalborg University
Niels Jernes Vej 14
DK-9220 Aalborg
DENMARK
Tel. + 45 96 35 97 99
Fax + 45 98 15 24 44
Email: kolda@3dgi.dk
Web site: http://www.3dgi.dk

Petr Soucek and Ales Cepek
Dept. of Mapping and Cartography, CTU Prague
Thakurova 7, 166 29 Prague
Prague
CZECK REPUBLIC
Tel. + 20 2 2435 4657
Fax + 420 2 2435 5419
Email: soucek@gama.fsv.cvut.cz, cepek@fsv.cvut.cz
Web site: http://www.gnu.org/software/gama/

ACKNOWLEDGEMENT

Support of the Grant Agency of the Czech Republic, project 205/03/0017, is highly appreciated.

Thanks to Jane Bengtsen for language corrections of the article.

APPENDIX A

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ONLY AN EXAMPLE SCHEMA CORRESPONDING TO UML DIAGRAM FROM ARTICLE AT FIG 2003 -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="FiniteElementsMesh">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="primitiveGeometry"/>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="GeometryNode">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="PointGeometry" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="PointGeometry" type="pointGeometry" abstract="true"/>
    <xs:element name="Location" abstract="true">
        <xs:complexType name="location" abstract="true"/>
    </xs:element>
    <xs:element name="Point" substitutionGroup="PointGeometry"/>
    <xs:element name="CameraPoint" substitutionGroup="PointGeometry">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="pointGeometry">
                    <xs:sequence>
                        <xs:element name="ATcameraNear">
                            <xs:simpleType><xs:restriction base="xs:float"/>xs:simpleType>
                        </xs:element>
                        <xs:element name="ATcameraFar">
                            <xs:simpleType><xs:restriction base="xs:float"/></xs:simpleType>
                        </xs:element>
                        <xs:element name="ATaspectRatio">
                            <xs:simpleType><xs:restriction base="xs:float"/></xs:simpleType>
                        </xs:element>
                        <xs:element name="AThorizFOV">
                            <xs:simpleType>xs:restriction base="xs:float"/></xs:simpleType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="pointGeometry" abstract="true">
        <xs:complexContent>
            <xs:extension base="primitiveGeometry">
                <xs:sequence>
                    <xs:element ref="Location"/>
                    <xs:element ref="GeometryNode" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="primitiveGeometry" abstract="true">
        <xs:sequence>
            <xs:element name="UnionOfPrimitiveGeometry" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="surfaceGeometry" abstract="true">
        <xs:complexContent>
            <xs:extension base="primitiveGeometry"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="linearGeometry" abstract="true">
        <xs:complexContent>
            <xs:extension base="primitiveGeometry"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="volumeGeometry" abstract="true">
        <xs:complexContent>
            <xs:extension base="primitiveGeometry"/>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```