A Tightly-Coupled XML Encoder-Decoder for 3d Data Transaction: A City-Modelling Scenario

Chengxi Bernad SIEW and Khairul Hafiz SHARKAWI, Malaysia

Key words: XML, Encoder, Decoder, Data Transaction

SUMMARY

Gaining interoperability for data transaction in Spatial Data Infrastructure (SDI) leads to increase of data volume has long been discussed in recent years. In order to solve large data volume arises due to XML self-describing capability, which is used in CityGML, a schema-aware encoder (CitySAC) is invented and achieved better compression ratio and require lesser time, compare to the state-of-the-art Lemper-Zipf-Markov (LZMA) algorithm. While geometric and semantic data is equally essential over the web services especially for analysis, the use case of this schema-aware encoder is defined. A decoder is created with proposed query interface, capable to perform direct query onto the compressed document. This paper discusses the schema-aware encoder background of development and the related works; as well as the showcase of tightly-coupled encoder-decoder over web service data transactions in city modeling scenario.

A Tightly-Coupled XML Encoder-Decoder for 3d Data Transaction: A City-Modelling Scenario

Chengxi Bernad SIEW and Khairul Hafiz SHARKAWI, Malaysia

1. INTRODUCTION

Sharing of spatial data over the internet has been popular over recent years, where different application-oriented SDIs are discussed in several researches. Moving from 2-dimensional to 3-dimensional extends the paradigm of backend infrastructure to dataset representation. Such extensions are seen in city modelling scenario, where Geography Mark-up Language (GML) is extended to City Geography Mark-up Language (CityGML) and different level-of-details are proposed (Kolbe, 2005). Whereas backend architecture of SDI has extended from Web Feature Service (WFS) to 3D layer represented by X3D via Web 3D Service (W3DS) (Basanow et. al, 2008). X3D supersedes Virtual Reality Mark-up Language (VRML). For visualization, delivering X3D for client-side could fulfil the requirements. Various compression schemes are discussed to efficiently encode geometrically and connectivity of 3D objects. However, semantics data embedded in CityGML is useful for analysis over the web, while not for efficient rendering of client-side visualization, both geometry and semantics data is useful for Web Processing Service (WPS) and sending raw CityGML without compression could require large bandwidth. This paper highlights the XML compression techniques in previous works and proposed a new technique highly suited large geometry inputs such as CityGML.

In this paper, the development of schema-aware encoder for CityGML and its results compared with dictionary and arithmetic compression modules such as 7-zip, WinZIP, InfoSet, etc are briefly discussed, then the following section discusses the development of javascript decoder and how the retrieval process could be done via code-on-demand in web application. The dataset that is used in this re-search is 3D Putrajaya city, managed by Malaysian Geospatial Data Infrastructure (MacGDI).

2. BACKGROUND AND RELATED WORKS

Various compression techniques had been discussed such as geometry compression related by Deering (1995), Isenburg (2000), Taubin (1998), whereas connectivity compression could be sub-divided into edge-based compression and vertex-based compression. Edge-based related techniques are discussed by Rossignac (1999) and Szymczak (2003) while vertex-based techniques are discussed by Alliez (2001) and Touma (1998). Not a single algorithm could serve all scenarios as shown in the review by Peng (2005). In Siew and Abdul Rahman (2013), a schema aware compressor is invented coupling with the LZMA compression module in the final stage. The development of the encoder is well discussed and could be implemented as a component in web services transaction. The main idea of the encoder is to allow "schemaawared" binary data transacted which could largely reduce file size up to 8% of original size. As compression schemes are well discussed in Isenburg (2013), this encoder adapts lossless, non-progressive, streaming and random access behavior. On the other hand, some discussed techniques are targeted specifically for XML, such as Infoset dictionary method (Oracle, 2005) and other general text compressors XMill (Liefke et al, 1999), XGrind (Tolani et al, 2002), XPRESS (Min et al, 2003), XComp (Li et al, 2003), XCQ (Ng et. al, 2007). However, none of them are suitable for solving the core problem in transmitting geometry and semantic data in a Document Object Model (DOM) form, which is embedded with equal importance for geometries and semantics information, e.g. CityGML. Some modifications are required for GIS 3D SDI use cases.

XMill (Liefke et al, 1999) implements split, group, and compress methodology and the GZip compression module placed at the last stage. Containers are prepared with structure domain and data domain, and each container is defined via default 8MB window size. To improve XMill, XGrind (Tolani et al, 2002) improved with query ability onto compressed document, while a structure of document is left without encoding. While the improvisation able to solve the query limitation which is not available in previous compressor, XGrind only adapts DTD of XML but not XSD which was introduced later in common XML document. On the other hand, XPRESS (Min et al, 2003) introduced reverse arithmetic encoder which divides the elements and content path into floating [0.0 to 1.0] interval. This concept was being used in other common arithmetic encoders. In addition, this encoder does not encode data types or the value of the elements. In the scenario of XComp (Li et al, 2003), it improves XMill by introducing more containers rather than structure containers, which are data-length container and dictionary container, while XCQ (Ng et. al, 2007) introduced indexing method via Pseudo-Depth-First strategy which builds a DTD tree. Fast Infoset (FI) uses ASN.1 notation which is currently the standard: ITU-T Rec. X.891 and ISO/IEC 24824-1.

While various SDI could be seen from around the world, such as INSPIRE (Infrastructure for Spatial Information in the European Community), Digital China Geospatial Framework (DCGF) (Lia et al, 2008), National Land Information System (NLIS) in Poland (Gazdzicki and Linsenbarth, 2004), and etc shows the importance of spatial data sharing for various requirements. Sharing of data using a common file format such as GML is seen in various researches (Christensen et. al, 2007). Extending 2-dimensional into 3-dimensional SDI initiatives are well discussed in Berlin-3D and Heidelberg-3D (Basanow et al, 2008), which uses CityGML and common data sharing format. Few others example that uses semantics and geometries in analysis are shown in, (Walenciak et. al, 2008) (Stollberg and Zipf, 2008) and (Lanig and Zipf, 2010). This paper aims to discuss the methodology or workflow that involved in developing the decoder for the prebuilt schema aware encoder CitySAC (Siew and Abdul Rahman, 2013) and how the web transaction could achieve interoperability meanwhile allowing binary data transaction within the web service using HTTP protocols.

3. THE SCHEMA-AWARE ENCODER

The flow of encoding pipeline follows stages discussed by Augeri (2007) shows the general stages on how information is transformed. Since CitySAC using dictionary encoding, all occurrences are indexed and stored. Furthermore, geometries are built in chunk of 65,000 face sets where each face represents a polygon in CityGML. The main idea of the encoder is to follow the XML original structure so query-able capability is retained and compressed content is retrievable. The encoder defines each representation as a symbol denoted in 16-bits.

The compression ratio is calculated with the following equation:

$$\mathrm{CR}_2 = \left(1 - \frac{\text{size of (compressed file)}}{\text{size of (original file)}}\right) \times 100\%$$

WinZip is used as the benchmark, and this encoder aim to perform better than direct 7-zip results in terms of timespan and compressed size and used 7-zip the final compression stage. It is worth mentioning that the encoder achieved a 35% to 50% better compression rate compare to WinZip (see Figure 3), and 20% to 30% better compression rate compare to direct 7-zip. In near-lossless scheme, better compression ratio is achieved (see Table 1 and Figure 1). The environment for ecoding is shown in Table 3. The time taken in seconds is shown in Figure 2.

Table 1. The compression ratio (near-lossless) with quantization option set to true

	Original (MB)	**Deflate zip (MB)**	**Fast-Infoset (MB)**	**LZMA (MB)**	**Bzip2 (MB)**	**CitySAC + Deflate (MB)**	**CitySAC + LZMA (MB)**
Commercial_ Building3C6.xml	0.836	0.086	0.469	0.073	0.079	**0.072 (91.39%)**	**0.056 (93.3%)**
National Audit.xml	8.594	0.948	4.132	0.696	0.869	**0.723 (91.59%)**	**0.561 (93.47%)**
Putrajaya_ Mosque.xml	10.928	1.56	4.81	0.992	1.226	**0.98 (91.03%)**	**0.782 (92.84%)**
Seri Gemilang Bride.xml	27.454	3.612	12.352	2.714	3.519	**2.59 (90.57%)**	**1.94 (92.93%)**

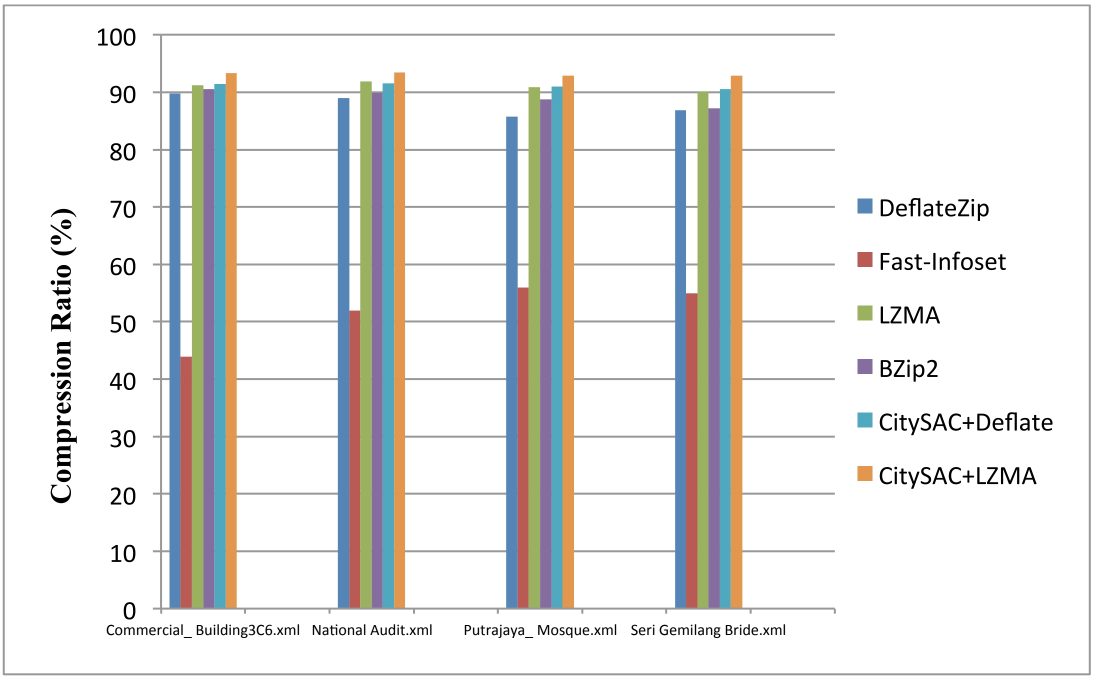

Figure 1. The compression ratio by different techniques based on original file size (Better for higher ratio)

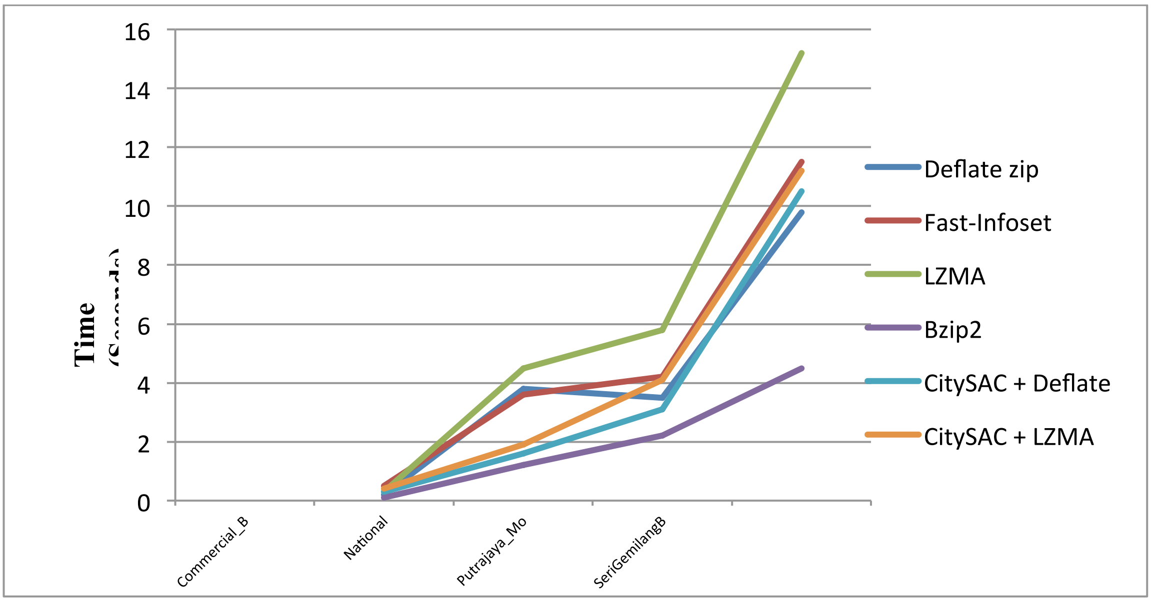

Figure 2. The time consumption for different techniques with default setting (Using Stopwatch library) (Better for lower reading)

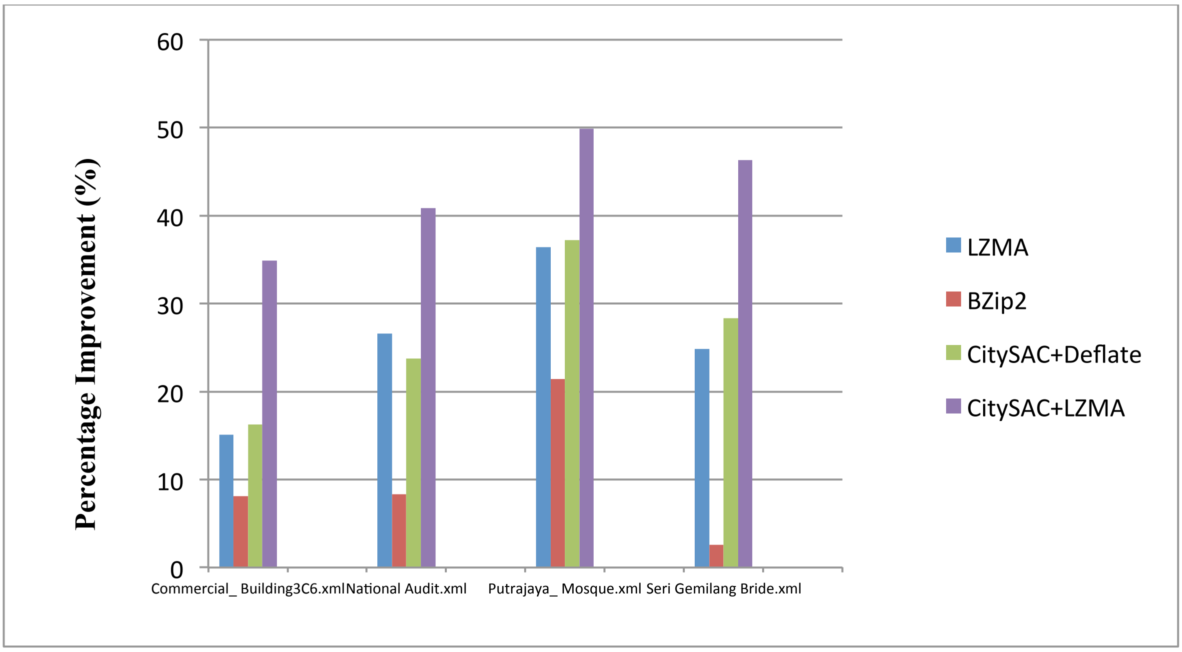

Figure 3. The improvement comparison between DEFLATE and other techniques (Better for higher percentage)

Techniques\Characteristics	**Query-able**	**Partial decompression**
Deflate	No	No
Bzip2	No	No
LZMA	No	No
Fast-Infoset	Yes	Yes
CitySAC	Yes	Yes

Table 2: The comparison of techniques adopting query-able and partial decompression capabilities

Processor	Intel Core i7-3610M 2.3Ghz
Memory	6GB DDR3

Hard Disk	1 TB HDD with 34.6GB free space on C:
Operating System	Windows 7 Professional on DOT.NET Framework 4.5

Table 3. The encoding environment

4. THE DECODER

The proposed encoder separated inputs into key XML components such elements, attributes, attribute values, values, URIs and geometries. Key components are identified along with the XPath scanning through the input data. While the data are encoded based on uniform 16-bit symbols, the information is in binary format and a specific decoder which is built in Javascript library is created to decode information both for geometries and semantics. Basically floating points or integers are treated as geometries while semantics are common textual data. The decoder which is built with code-on-demand LZMA or DEFLATE decoder will then decode information and parse to the key components container. While the entire process depends on the query requirements, the information flow basically follows the process depicted in Figure 4. Query interface towards the encoded data is depicted as the modules in Figure 4. With the capability of Javascript to read BigEndian and little-endian binary format, interoperability is achieved on top of binary data. By employing tightly-coupled encoder decoder architecture in web service, it could of course allow small data transfer which could highly reduce the bandwidth consumption and time consumption.

The decoder is built with a binary-reader.js (See jDataView) and lzma.js (See LZMA-JS) which are openly available. With these javascripts the decoder able to decode the binary data and information is retrieved via the modules that allow user to decode fully back to original CityGML file or query the XML tag on-the-fly. Figure 4 shows the workflow for different modules in the decoder. The decoder shows how information could be retrieved by modules that are written in javascript.

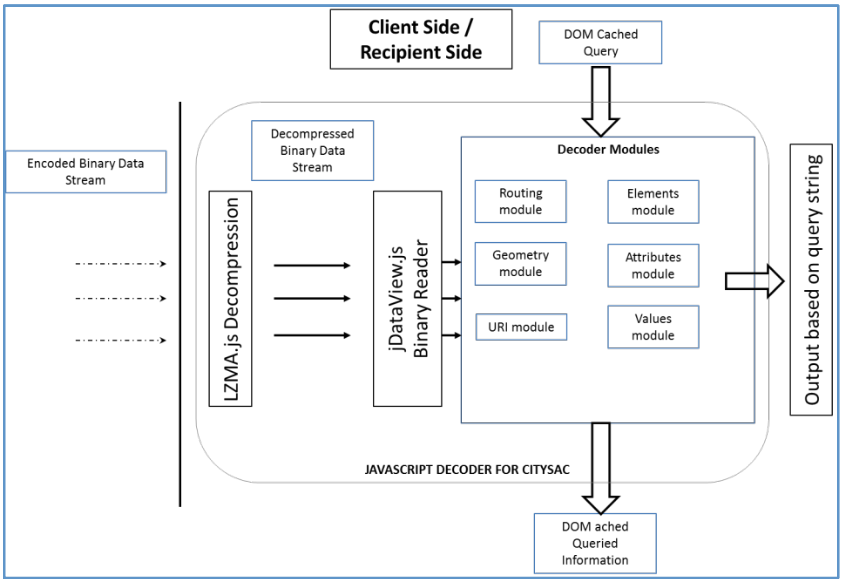

Figure 4. The javascript decoder workflow in modules (Siew and Abdul Rahman, 2013).

The encoded binary stream is retrieved and will be decompressed by on demand lzma.js and decoded stream will be read by binary reader supported by jDataView.js. The reader will then populate the related information in different modules which later on parsed the output for client side applications. The default output of the decoder generates CityGML data. However, using customized module such as routing module, specified data output could be generated such that only geometry array or URI identifier. The cached could then be retrieved as a Document Object Model (DOM).

5. IMPLEMENTATION AND DISCUSSIONS

The implementation of the Javascript decoder in browser script realise the tightly-coupled encoder-decoder architecture, while ensure data interoperability outputs. This approach is a step forward to realize binary stream between client and server without platform-dependency at client side and allow small data size transactions. Though thin-client does not require CityGML as the data input, however the decoder provides advantages for medium-client and thick-client usage, as well as within web services transactions.

For city-modelling scenario, buildings are retrieved based on query such as the Key Value Pair (KVP) formatting. Data from 3D database generates CityGML, transformed into binary stream via the aforementioned encoder and send to receiver. Binary data will then be transferred and be decoded via lzma.js and then be parsed to key components containers and then generates useful information either to CityGML format, or X3D for visualization, or geometry information that built in the decoder via query interface. This process applies to web services where the query generates CityGML at server side, encoded and send to relevant web services for data processing. Figure 5 depicted the scenario where applying encoder-decoder coupling in web services allow minimal data transaction while maintaining interoperability.

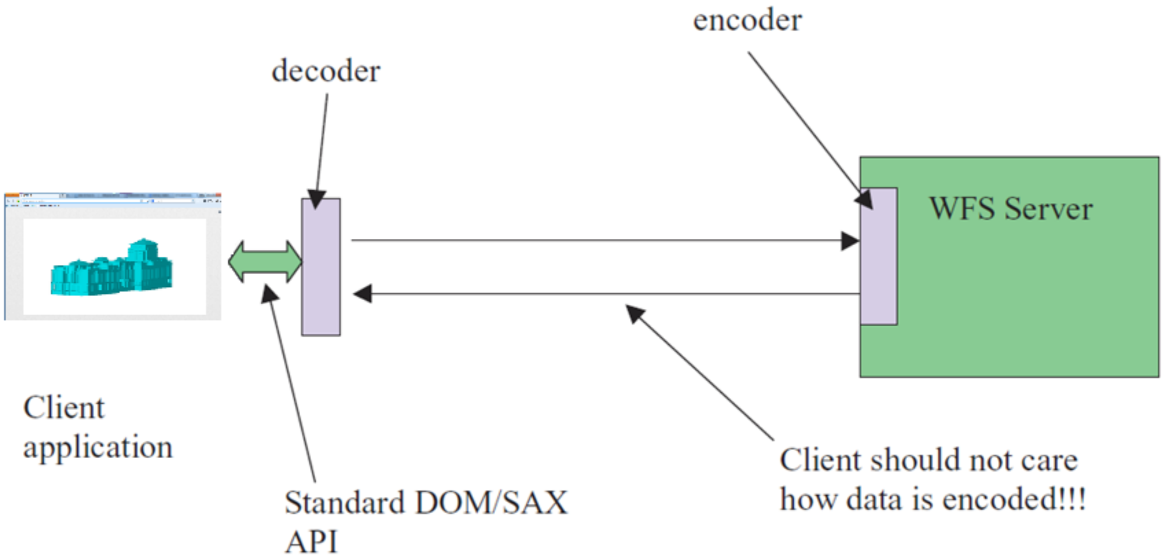

Figure 5. Encoder-decoder coupling in web services for efficient transmission

6. CONCLUSIONS

In this paper we discussed the performance of the encoder and the usage of the decoder with the query interface to retrieve information from the compressed document. The decoder is developed using Javascript browser language and open source Javascript library. The output of the decoded information is readable therefore interoperability is achieved on top of binary streamed data. We also demonstrate the information flow of the city-modelling application for Putrajaya 3D Corporation, Malaysia. The compressed document is compared with the state-of-the-art compressor and the average query time on geometry for a indexed face-set took around 16ms. In this experiment, it shows that XML compression technique is an alternative approach towards SDI data transaction as common gzip is insufficient due to the lacking of query capability.

REFERENCES

Alliez P., Desburn M., 2001, Valence-Driven Connectivity Encoding for 3D Meshes, Eurographics '01

Augeri Christophe J., Mullins Barry E., Bulutonglu Dursun A., Baldwin Rusty O., Baird III Leemon C., 2001, An Analysis of XML Compression Efficiency, EXPCS, 13-14 June. 2007, San Diego, CA. 978-1-59593-751-3/07/06

ASN.1: Fast infoset, 2005, Series X: Data Networks, Open System Communica-tions And Security, ITU-T X.891, Telecommunication Standardization Sector Of ITU

Basanow J, Pascal N, Neubauer S, Schilling A, Zipf A, 2008. Towards 3D Spatial Data Infrastructures (3D-SDI) based on open standards – experiences, re-sults and future issues, Advances in 3D Geoinformation Systems (2008) Springer, pp 65-86

Christensen A. Friis, Lutz M., Ostlander N., Bernard L., 2007, Designing Service Architectures for Distributed Geoprocessing, Transactions in GIS 11(6), pg. 799-817

Deering M., 1995, Geometry Compression, SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques

Gazdzicki J, Linsenbarth A, (2004), 10th EC GI & GIS Workshop, ESDI State of the Art, Warsaw, Poland

Isenburg M., Snoeyin J., 2000, Spirale Reversi: Reverse Decoding of EdgeBreaker Encoding, Proceeding of 12th Canadian Conference on Computational Geometry

Isenburg Martin, 2013, LAZzip: Lossless compression of LiDAR data, ASPRS Journal

JDataview, https://github.com/gmarty/jsDataView. Accessed: 2013-Nov-15

Kolbe T. H., Gerhard K., Nagel C., Stadler A., 2008, 3D Geodatabase Berlin ver-sion 2.0.1a, Fachgebiet Methodik der Geoinformationstechnik der TU Ber-lin, 2008.

Lanig Sandra, Zipf Alexander, 2010, Proposal for a Web Processing Services (WPS) Application Profile for 3D Processing Analysis, 2010 Second Inter-national Conference on Advanced Geographic Information Systems, Ap-plications, and Services, IEEE 978-0-7695-3951-5/10, pg 117-122

Lia P, Lan W, Xiao X, (2008), Sdi In China：Progress And Issues, The Interna-tional Archives of the Photogrammetry, Remote Sensing and Spatial In-formation Sciences. Vol. XXXVII. Part B4. Beijing

Liefke H. and D. Suciu, 1999. XMill: An Efficient Compressor for XML Data. http://sourceforge.net/projects/xmill Accessed: 05-May-2012
Li, Weimin. Xcomp: An XML compression tool. Diss. University of Waterloo, School of Computer Science, 2003.

LZMAJS, https://github.com/nmrugg/LZMA-JS, Accessed: 2013-Nov-15

Min J., M. Park and C. Chung., 2003. XPRESS: A Queriable Compression for XML Data, Proceedings of the 2003 ACM SIGMOD International Confer-ence on Management of Data, 2003

Ng W., Lam W.Y, Wood P.T., Levene M., 2006. XCQ: A queriable XML compression system, Springer-Verlag London Limited 2006, Knowl Inf Syst (2006) 10(4): 421–452

Oracle Sun, Fast Infoset Documentation,
http://java.sun.com/developer/technicalArticles/xml/fastinfoset/ Accessed: 01-March-2011

OGC, 2010. Draft for Candidate OpenGIS® Web 3D Service Interface Standard, Version: 0.4.0, OGC 09-104r1 pp 95

Peng Jingliang, Chang-Su Kim, C.-C. Jay Kuo, 2005, Technologies for 3D mesh compression: A survey, J. Vis. Commun. Image R. 16 (2005) pp. 688–733

Rossignac Jarek, 1999, Edgebreaker: Connectivity compression for triangle mesh-es, GVU Technical Report GIT-GVU-98-35

Shannon C. Communication in the presence of noise. Proceedings of the IRE, 37(1):10-21, 1949

Siew C Bernad, Alias A Rahman, 2013, A schema-aware encoder for putrajaya 3d, Urban and Regional Data Management, UDMS Annual 2013 - Pro-ceedings of the Urban Data Management Society Symposium 2013 , pp. 181-190

Stollberg Beate, Zipf Alexander, 2007, OGC Web Processing Service Interface for Web Service Orchestration, Springer-Verlag

Szymczak A., J. Rossignac, Wrap and zip decompression of the connectivity of triangle meshes compressed with Edgebreaker, Comput. Geom. 14 (1-3) (1999) 119–135

Taubin, G., Horn, W., Lazarus, F., Rossignac, J. (1998), Geometry coding and VRML, Proceedings of The IEEE, 86(6)

Tolani P. and J. Harista., 2002, XGrind: A Query-friendly XML Compressor, IN ICDE, pp 225-234

Walenciak Georg, Stollberg Beate, Neubauer Steffen, Zipf Alexander, 2008, Ex-tending Spatial Data Infrastructure 3D by Geoprocessing Functionality.

BIOGRAPHICAL NOTES

Siew Chengxi Bernad, obtained a Bsc Degree in Geoinformatics in 2009 from Universiti Teknologi Malaysia (UTM), Skudai, Johor. He is a professional programmer in JAVA and C# programming language, while currently highly involving in HTML5 and Javascript. He worked as an IT Manager in ABS Innovations Sdn Bhd and a Chief Technical in Effisys Technology, Malaysia. He has been involving in developing various web applications and mobile applications. Currently, he is a PhD researcher in 3D GIS Research Group in the Faculty of Geoinformation and Real Estate, UTM. His current research interests are in the area of distributed web environment for 3D GIS and compression techniques for 3D datasets.

Khairul Hafiz Sharkawi, obtained an MSc degree in Geoinformatics in 2010 and a BSc degree in Geoinformatics in 2006 from Universiti Teknologi Malaysia. He is currently a PhD researcher with the 3D GIS group of the Faculty of Geoinformation and Real Estate at the Universiti Teknologi Malaysia (UTM) in Malaysia.

CONTACTS

Siew Chengxi Bernad
3D GIS Research Lab
Faculty of Geoinformation and Real Estate
Universiti Teknologi Malaysia
81310 Johor Bahru
MALAYSIA
Email: cbsiew2@live.utm.my
Contact: +6017-8888460

Khairul Hafiz Sharkawi
3D GIS Research Lab
Faculty of Geoinformation and Real Estate
Universiti Teknologi Malaysia
81310 Johor Bahru
MALAYSIA
Email: hafiz.sharkawi@gmail.com
Contact: +6016-8954818