Application of Schema-Aware Encoder-Decoder in 3D City Inventory Management

Siew, C. B. and Abdul Rahman, A.
Faculty of Geoinformation and Real Estate, Universiti Teknologi Malaysia, Johor Bahru, Malaysia
E-mail: bernad@bernadsiew.com

Abstract
The use cases of Spatial Data Infrastructure (SDI) had been extensively discussed in various researches from disaster management, city inventory management and etc. The realization of spatial data sharing within national agencies and stakeholders is due to the formation of data standardizations for interoperability, and service oriented architecture (SOA). Extension of such technology could nevertheless be applied in three-dimensional (3D) spatial data sharing. However, certain issues need to be addressed and solved, such as visualization pipeline and analysis pipeline for 3D spatial data in SOA. This paper discusses existing 3D SDI frameworks, and how an efficient encoder-decoder coupling method could be one solution for analysis pipeline in which both geometric and semantic information should be present. The framework is then integrated into building inventory management and 3D buildings are presented at client side for property tax appraisal exercise.

1 Introduction

The use of city models in various applications is becoming increasingly important, as can be seen in various developments of 3D SDIs (Basanow et al., 2008, Neubauer and Zipf, 2007 and Oude Elberink et al., 2013). Another related work in this aspect mentioned by Kolbe (2008) that city models are especially important for presentation of business locations and urban development that is required by public and private sectors, such as providing the platform that supports the process from civic participation to decision-making and policy-formulation. Enabling 3D spatial data sharing leveraging SOA could largely improve the decision making process in top management who requires spatial data. It can be clearly seen with the establishment of SDI in various countries (e.g. INSPIRE (Infrastructure for Spatial Information in the European Community), Digital China Geospatial Framework (DCGF) (Li et al., 2008), National Land Information System (NLIS) in Poland (Gazdzicki and Linsenbarth, 2004) - shows the importance of spatial data sharing for various requirements. In the aforementioned 3D SDI architectures, encoding 3D city model in CityGML is commonly used as it could describe complex urban environment, that is, representations of 3D analysis and visualization. Also, Choosumrong et al., (2014) described how service oriented architecture (SOA) being applied for the establishment of dynamic routing as web service. Such applications or initiatives are part of the global initiatives shown in the increasing trends and requirements in spatial data sharing. The continuous growing CityGML specifications could be seen useful to serve broader applications and more detailed information in terms of Level-of-Details (LoDs). With the robust semantic representations given in CityGML, Cagdas (2013) has proposed an application domain extension to CityGML for immovable property taxation as previously being discussed by Dillinger (1991) on property tax system requirements. These two previous works clearly illustrated the requirements for a complete and accurate inventory – geometric and semantic information in property tax system. Hence, CityGML fits in this use case. However, it is commonly accepted that transferring CityGML to client side is impractical due to the nature of XML datasets' drawbacks (see Siew and Abdul Rahman, 2013), which is the repeating readable elements that increases transmission bandwidth and prone to human intervention. In this paper, Section 2 examines some existing SDI frameworks and background study. The development of schema-aware encoder for CityGML and its results compared with dictionary and arithmetic compression modules such as LZMA, WinZIP, InfoSet, etc are then briefly discussed in Section 3. It is then implemented with the Javascript decoder at client side described in Section 4 for property tax appraisal exercise. A framework is presented and future works are also discussed in the final section. The dataset used in this research comes from 3D

Putrajaya city project, which is managed by the Malaysian Geospatial Data Infrastructure (MaCGDI) agency.

2. Background and Related Works

Previous local SDI frameworks are built mainly on web services as well as Open Geospatial Consortium (OGC) Web Services (OWS). Galip (2007) showcased a SDI framework (Figure 1) for real time data grid utilizing binary XML as middleware data transaction.

This example showed the requirements for having GIS over distributed environment; compare to existing desktop GIS solutions. Utilizing web services and OWS allowed interoperability in data transaction within different organizations. The similar workaround was discussed in Basanow et al., (2008) where general OWS and OpenLS is developed for local SDI for navigation as depicted in Figure 2. Other scenarios in using OWS for SDI were discussed in Christensen et al., (2007) and Gong et al., (2009).

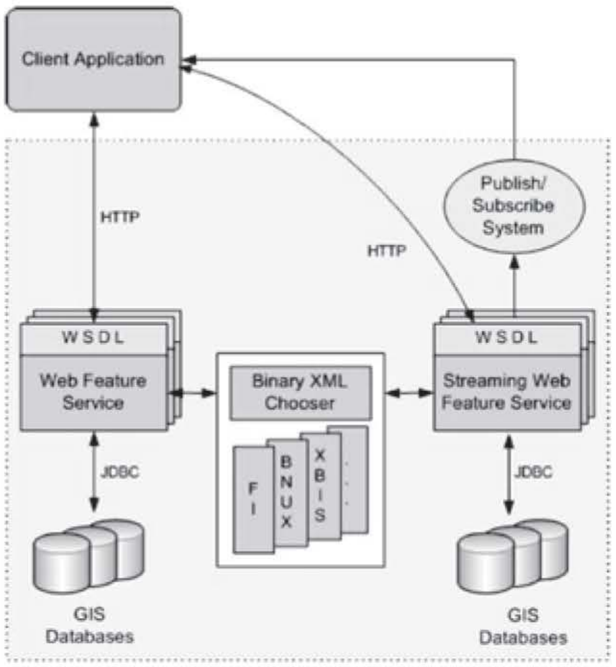

Figure 1: The SDI framework supporting real time data grid incorporating binary XML framework (Galip, 2007)

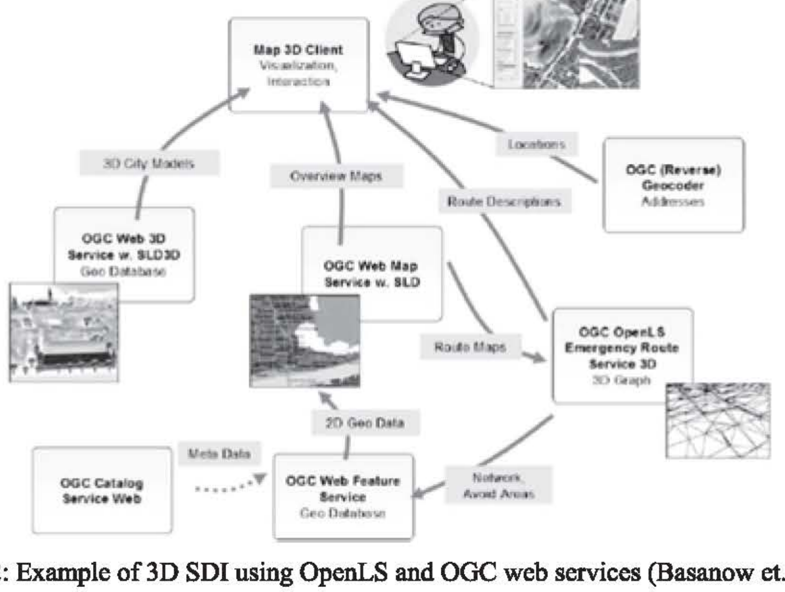

Figure 2: Example of 3D SDI using OpenLS and OGC web services (Basanow et. al, 2008)

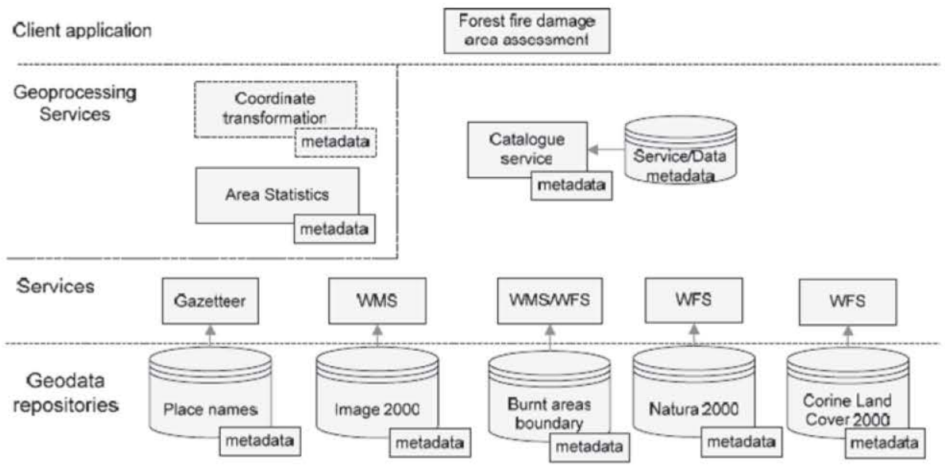

Figure 3: Example of SDI for forest fire damage assessment (Christensen et. al, 2007)

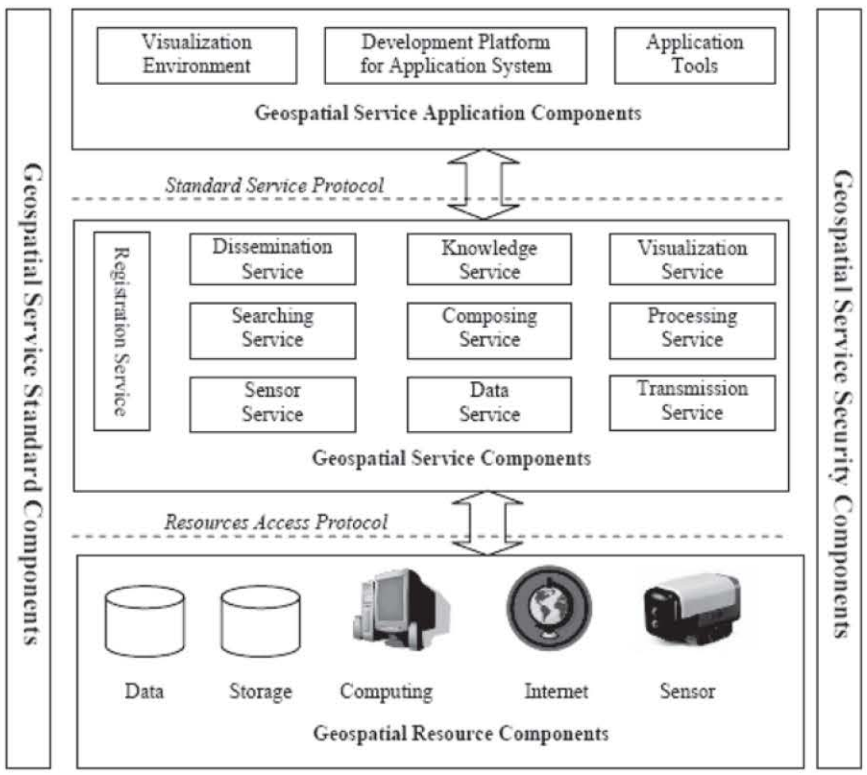
Visualization Environment
Development Platform for Application System
Application Tools
Geospatial Service Application Components
Standard Service Protocol
Registration Service
Dissemination Service
Knowledge Service
Visualization Service
Searching Service
Composing Service
Processing Service
Sensor Service
Data Service
Transmission Service
Geospatial Service Components
Resources Access Protocol
Data
Storage
Computing
Internet
Sensor
Geospatial Resource Components
Geospatial Service Standard Components
Geospatial Service Security Components

Figure 4: Geospatial services components (Gong et al, 2009)

Christensen et al., (2007) as depicted in Figure 3 uses OWS to establish a SDI framework managing forest fire damage assessment while Gong et al., (2009) focused on establishing a framework for geospatial information dissemination on the web (see Figure 4). CityGML is based on GML (Geography Mark-up Language), which is an XML standard – a popular W3C standard in data exchange and sharing usage in computing systems due to its readable and self-describing benefits. However, given the cost involved in storing and transferring the schema over distributed networks, transmitting models in CityGML format is considered impractical, especially when the city model is used for visualization purposes over the distributed environment. The use of X3D for city model representation in Web 3D Services (W3DS) (OGC, 2010) is then utilized. X3D is also a self-describing format in XML standard, one that is very verbose. On the other hand, dealing with semantic data that is required by analysis for orchestration in Web Processing Service (WPS), a data model with semantics is necessary.

To deal with these requirements, transmitting CityGML in large datasets should be compressed; therefore, achieving efficiency in terms of data size and compression time is essential. Various compression techniques had been discussed such as geometry compression and connectivity compression. While in recent researches, XML compression is getting popular as more web services are utilizing XML schema as standards in data transaction. To achieve better efficiency, XML compression seems viable as this could improve the current compression scheme where some results are shown in Section 3. Published techniques such as Infoset dictionary method (Oracle, 2005) and other general text compressors XMill (Liefke and Suciu, 1999), XGrind (Tolani and Harista, 2002), XPRESS (Min et al., 2003), XComp (Li et al., 2003), XCQ (Ng et al., 2006) are targeting specifically for XML datasets, 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. 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 "schema-aware" 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. XMill (Liefke and Suciu, 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. The workflow is shown in Figure 5. To improve XMill, XGrind (Tolani and Harista, 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., 2006) 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.

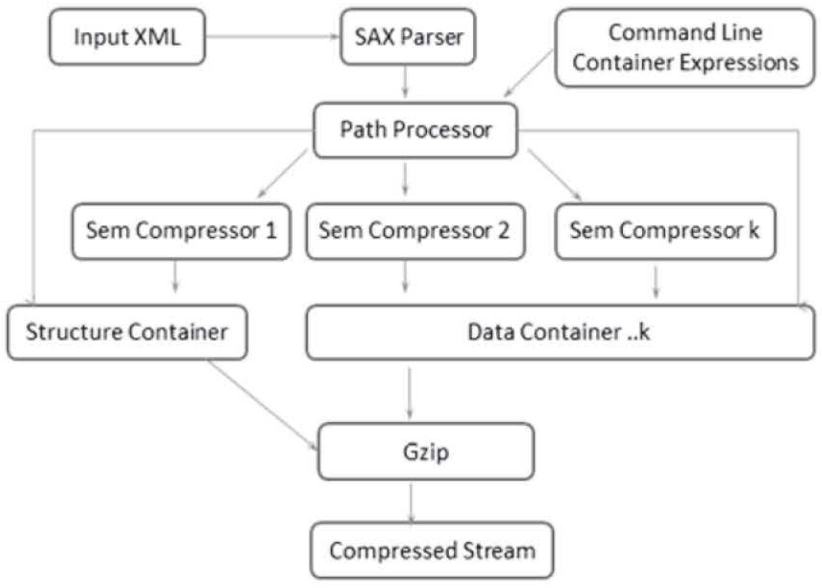

Figure 5: XMill Workflow (Liefke et al, 1999)

3. The Encoding Engine

The requirements for encoding CityGML in 3D SDI are such as large XML data is inefficient for both geometric and semantic information, existing encoding method output could not be query-able and it is not suitable for the schema of CityGML, as well as the development of HTML5 allows binary data transfer in platform-independent way while maintaining interoperability (Siew and Abdul Rahman, 2013). Reviewing the existing techniques XMill (Liefke and Suciu, 1999), XGrind (Tolani and Harista, 2002), XPRESS (Min et al., 2003), XComp (Li et al., 2003), XSLC (Wang et al., 2005), and XCQ (Ng et al., 2006), the in-house developed encoder employs 16-bit encoding scheme, and embedded with refined XML compression technique which is named CityGML Schema Aware Compressor (CitySAC). The concept is then tested with 3D buildings developed from previous city modelling projects at Putrajaya, the administration city of Malaysia. In addition, the encoder achieved better in compression ratio compared to state-of-the-art LZMA-alone compressor. Furthermore, in most of our case studies, the in-house developed encoder compress faster than LZMA-alone compressor using moderate performance setting as shown in Table 2. 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 architecture of the encoder and how a query-able scheme could be encoded. The prototype is developed using C# wrapped with XMLReader and XMLNode-identifier.

Figure 8 shows the compression engine which comprises several components such as XMLParser, XSD builder, Entropy Module, Dictionary builder, the Parent-Child Identifier (PCI), Scaled Geometry Constructor (SGC) and the compression module. The parser is capable of parsing XML information which is a wrapper of the built-in XMLReader in C# library. The parser parses information such as XML nodes, depths, namespace, value and etc. It then scans the document and retrieves each node's information, then allows navigating through forward and backward to retrieve intermediate information such as attributes and etc. XSD Builder builds a schema reference which would be the main reference coupled with dictionary. This builder runs if the XML document accompanied by a XSD. If the XSD is not present, then XSD will be built by scanning the XML document once to parse the relevant information into dictionary. Entropy module generates entropy by using the value parsed out by the XMLParser, and compared with the same type. This module is to ensure the representations always within the bounding of the 16-bit representations. This module also helps to determine if a higher number of representation is required, which allows extension of the current encoder for the next phase of development. Dictionary builder uses HashSet to scan unique value parsed by XMLParser and store into different containers as shown in Figure 6. Dictionary data only store unique value for each data type. Dictionary builder stores if the value is unique, and produce an index to the value if the value is repeating. This module maps the data input and largely reduce repeated occurrences. Parent-child Identifier (PCI) uses the URI information given by the document, and constructs the pseudo-tree relation with the nested child.

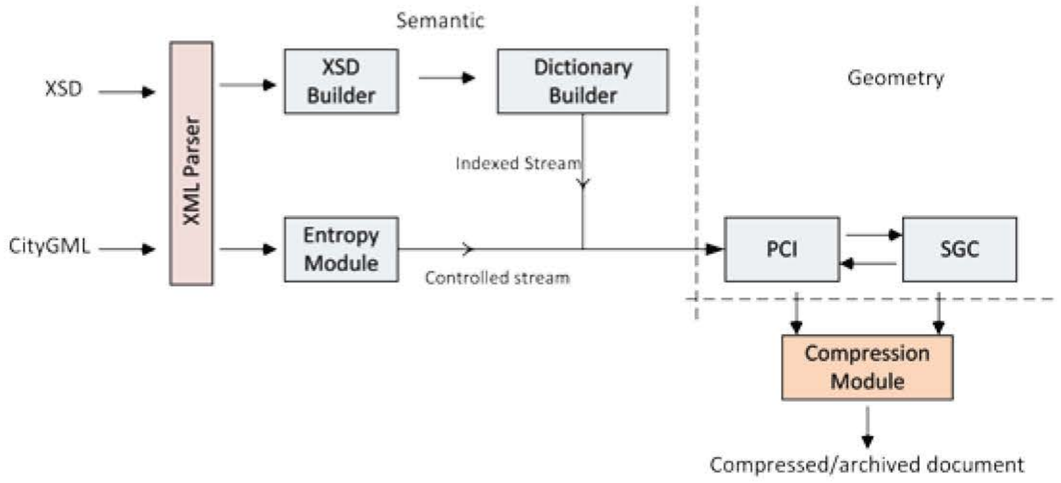

Figure 6: The encoding workflow in CitySAC

The children are examined by using given XML-depth information generated by XMLParser. This information is essential for constructing the parent and child to maintain the topological relationship within the nested polygon such as Composite Surface. With this PCI tree, a catalog is generated that could be used as client-side retrieval for partial decompression. While having PCI tree on the other side, SGC performs quantization for the geometries and the difference of the relative value generated from K-Face-Mean algorithm is stored. The relationship between geometries and the face-sets are constructed in this constructor, which then allows partial decompression by client-side retrieval. For example, if face-sets of 65001 and 65002 are only required from query, only both of these containers are decompressed rather than the entire data. The compression ratio is calculated with the following equation:

$$CR_2 = \left(1 - \frac{\text{size of (compressed file)}}{\text{size of (original file)}}\right) \times 100\%$$

Equation1

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, and 20% to 30% better compression rate compare to direct 7-zip (See Table 3). In near-lossless scheme, better compression ratio is achieved (see Table 1). The processor for encoding is Intel Core i7-3610M 2.3Ghz with 6GB DDR3 memory and running on DOT NET Framework 4.0 Windows 7.

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%)**

Table 2: The time consumption for different techniques with de-fault setting (Using Stopwatch library)

Datasets	**Original (MB)**	**Deflate zip (MB)**	**Fast-InfoSet**	**LZMA**	**Bzip2**	**CitySAC +Deflate (MB)**	**CitySAC + LZMA (MB)**
			(MB)	**(MB)**	**(MB)**		
Commercial_Building3C6.xml	0.836	0.2	0.3	0.6	0.1	**0.5**	**0.5**
National Audit.xml	8.594	1.9	1.0	6.0	0.8	**2.1**	**2.5**
Putrajaya_Mosque.xml	10.928	2.1	1.5	8.2	1.2	**2.6**	**3.2**
Seri Gemilang Bride.xml	27.454	4.9	2	23.6	2.3	**6.8**	**7.6**

Table 3: The improvement comparison between CitySAC and LZMA

Compressed Document	**LZMA (MB)**	**CitySAC + LZMA (MB)**
Commercial_Building3C6	0.073	0.056 (23.29%)
National Audit	0.696	0.561 (19.83%)
Putrajaya_Mosque	0.992	0.782 (21.17%)
Seri Gemilang Bride	2.714	1.94 (28.52%)

4. The Client-side Application

Discovery of all properties subject to taxation is required in property tax system, with the application domain extension proposal by Cagdas (2013), such system will require full CityGML content at client side. According to Cagdas (2013), a well-functioning cadastral system provides the basis for the property tax inventories. On top of that, information which is lacking but is required by property tax administration such as 1) physical, economic, environmental data regarding property units with buildings which is particularly needed for appraisal processes, and 2) the spatial representation of 3D property units which allows tax administration to verify taxable objects have been included in the inventories. These requirements fit the purpose of geo-semantic framework development as information from various aspects could be retrieved at a common platform development on the basis of open web services. Figure 7 shows the framework of how encoder could be used as intermediary services and with the decoder developed in Javascript, a platform independent script for client side, intermediary encoder service could be coupled within OWS. As far as the encoding engine is concerned, taxation extension for CityGML which is a XML-based content could be encoded into compressed format and be queried over by decoder on-demand. Information is retrieved from repositories via query layer through WFS and WMS. Base map is retrieved via WMS through Mapping Agency while detail features (2D) are retrieved from land office for lot information via WFS. While this information is required with or without requiring encoding service, it could be directly feed into client application. On the other hand, 3D buildings are retrieved from 3DDB with or without cadastral information from cadastral database. 3DDB will generate CityGML as output via CitySAC service and data encoding will be done. With on-demand solution at client side, encoded document will then be decoded into respective container to generate scene via X3D and retrieve semantic information at client side. Query results will be all information regarding a particular building differentiated in Level-of-details (LoD). Such approach could ensure offline usage for tax administrator even for certain location without data access.

5. Concluding Remarks

In this paper we showcase an application within 3D SDI requiring binary encoding and decoding engine, while maintaining interoperability for client side via javascript decoder (Siew and Abdul Rahman, 2014). We presented a case study for the usage of full CityGML information in property tax administration. A workflow is also demonstrated with performance evaluation for client-server procedure. By applying the encoder-decoder system client able to view, retrieve and analyze information with respect to tax appraisal exercise. The procedure could improve the efficiency of tax administrator in detecting new building inventories as well as incorporating change detection algorithm on top of the current system.

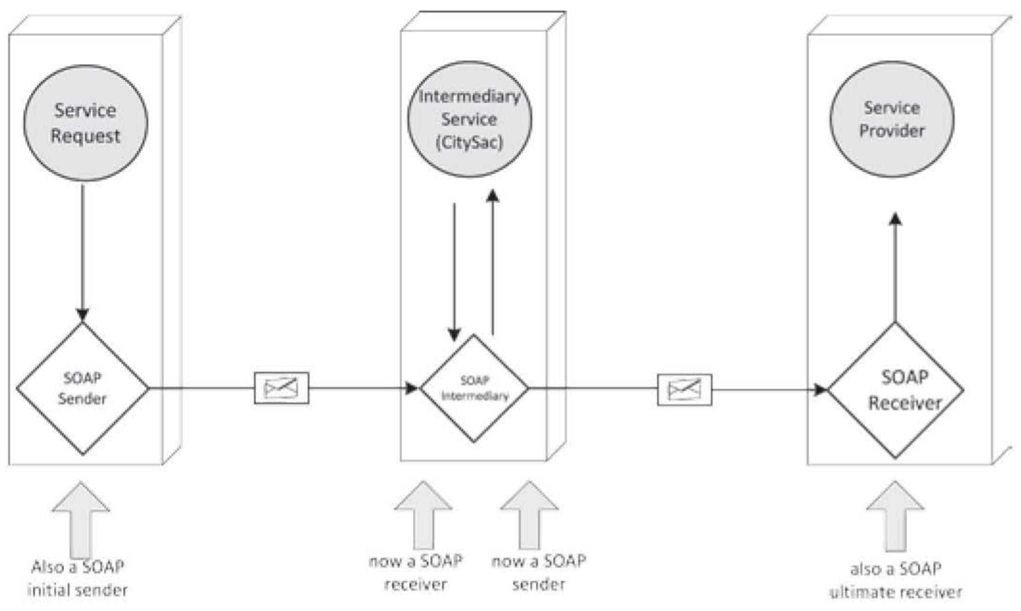

Figure 7: The intermediary service for encoding service

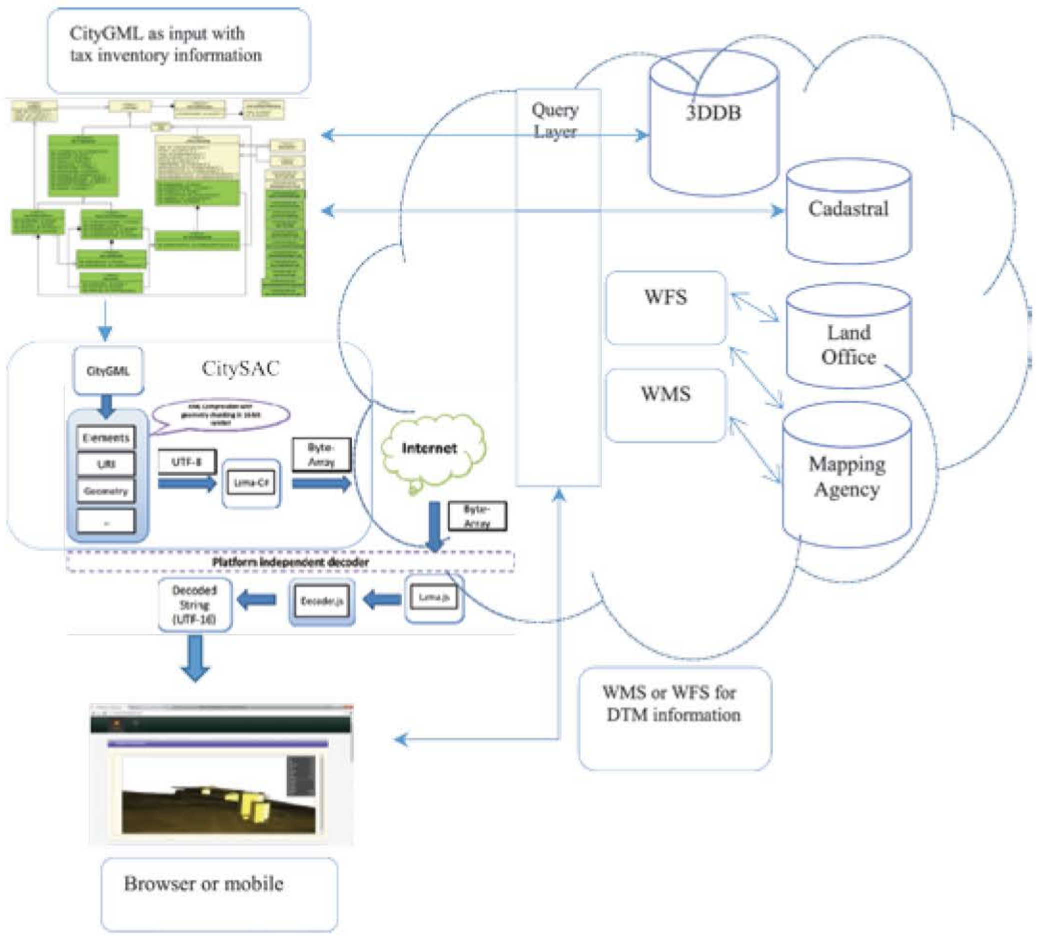

Figure 8: The framework integrating encoder-decoder engine

This application could be extended to network protocol embedding the binary schema as proposed by this encoder. Furthermore, the encoding process could be improved via semantic ontology analysis where only relevant semantic information could be retrieved for client side instead of full 3D building information in any scenario.

References

Basanow, J., Pascal, N., Neubauer, S., Schilling, A. and Zipf, A., 2008, Towards 3D Spatial Data Infrastructures (3D-SDI) Based on Open Standards – Experiences, Results and Future Issues, *Advances in 3D Geoinformation Systems (2008) Springer*, 65-86

Cagdas, V., 2013, An Application Domain Extension to CityGML for Immovable Property Taxation: A Turkish Case Study, *International Journal of Applied Earth Observation and Geoinformation*. 21 (2013), Elsevier, 545-555

Choosumrong, S., Raghavan, V., Delucchi, L., Yoshida, D., and Vinayaraj, P., 2014, Implementation of Dynamic Routing as a Web Service for Emergency Routing Decision Planning, *International Journal of Geoinformatics*, Vol. 10, No. 2, Jun, 2014, 13-20

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

Dillinger, W., 1991, Urban Property Tax Reform, Guidelines and Recommendations. Working Paper. World Bank, Washington, DC.

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

Galip, A., 2007, Service Oriented Architecture for Geographic Information Systems Supporting Real Time Data Grids, PhD Dissertation, Department of Computer Science Indiana University.

Gong, J., Huayi, W., Wenxiu, G., Peng, Y. and Xinyan, Z., 2009, Geospatial Service Web, Geospatial Technology for Earth Observation, Springer Science, DOI 10.1007/978-1-4419-0050-0_13

Isenburg, M., 2013, LAZzip: Lossless Compression of LiDAR Data, *ASPRS Journal*.

Kolbe, T. H., König, G., Nagel, C. and Stadler, A., 2008, 3D Geodatabase Berlin ver-sion 2.0.1a, Fachgebiet Methodik der Geoinformationstechnik der TU Berlin.

Li, P. A., Lan, W. and Xiao, X., 2008, SDI In China : Progress And Issues, The International Archives of the Photogrammetry, *Remote Sensing and Spatial Information Sciences*. Vol. XXXVII. Part B4. Beijing

Liefke, H., and Suciu, D., 1999, XMill: An Efficient Compressor for XML Data. http://sourceforge.net/projects/xmill [Accessed: 05-May-2012]

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

Neubauer, S. and Zipf, A., 2007, Suggestions for Extending the OGC Styled Layer Descriptor (SLD) Specification into 3D–Towards Visualization Rules for 3D City Models. *Applied Sciences*, 133.

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

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

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

Oude Elberink, S., Stoter, J., Ledoux, H. and Commandeur, T., 2013, Generation of a National Virtual 3D City and Landscape Model for the Netherlands. In Jaarverslag 2012 Nederlandse Commissie voor Geodesie, KNAW-NCG, 2013, 41-56.

Siew, C. B. and Abdul Rahman, A., 2013, A Schema-Aware Encoder for Putrajaya 3D, Urban and Regional Data Management, UDMS Annual 2013. *Proceedings of the Urban Data Management Society Symposium*.181-190.

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

Wang T.J., Gao J., Yang D.Q., Tang S.W. and Liu Y.F. (2005). XPath Evaluation Oriented XML Data Stream Compression, Journal of Software Vol. 16, No. 5, 1000-9825/2005/16(05)0869: 869-877