Measuring Rapid Streaming Delivery of Synthetic Environments

Greg Peele, Andrew Bertrand, Haris Javaid, Michael Shin
Applied Research Associates, Inc.
Orlando, FL
gpeele@ara.com, abertrand@ara.com, hjavaid@ara.com, mshin@ara.com

Julio de la Cruz
ARL HRED ATSD
Orlando, FL
julio.a.delacruz4.civ@mail.mil

ABSTRACT

The rise of cloud computing, the wide deployment of mobile device platforms, and the convergence of live, virtual, constructive, and gaming domains require a major shift in synthetic environment exchange and delivery. Traditionally, synthetic environment exchange involves ad-hoc database files in specialized application-targeted formats. That approach does not scale to the whole-earth management necessary to deploy environment content to users across all domains at the point of need. Current web technologies standardized by the Open Geospatial Consortium (OGC) move us closer to the goal of streaming environment delivery. However, these XML-based protocols present bandwidth and latency challenges, particularly for mobile devices, and they lack a caching mechanism for limited or unreliable networks. Furthermore, the current standards lack a unified view of the full environment—five disjoint protocols are required to deliver content to a game engine renderer.

Several ongoing development and standardization efforts aim toward comprehensive synthetic environments designed for simulation and runtime performance requirements. For example, the OGC CDB draft standard defines a binary file-based repository for a multi-resolution runtime environment. The Khronos Group GL Transmission Format (glTF) optimizes binary transmission of streamlined visual content. The Army Research Lab's Layered Terrain Format (LTF) provides extensible compressed streaming of a comprehensive environment protocol backed by a file cache. To determine the performance gains of these newer approaches over conventional web standards, we need a common methodology for analyzing streaming repository performance. In this paper, we document a methodology to quantify bandwidth, storage, and latency metrics in a consistent manner using open source tools. We prove out this methodology to measure the baseline performance of existing OGC and de facto industry standards with real-world geospatial data. Using this methodology and control group, in future experiments we can properly determine the gains offered by new developments in the synthetic environment domain.

ABOUT THE AUTHORS

Greg Peele is a Senior Software Engineer and Project Manager at the Virtual Heroes Division of Applied Research Associates. Mr. Peele has worked for more than ten years developing innovative terrain generation solutions for the military training community.

Julio de la Cruz is the Chief Engineer for Synthetic Environments Research of the Simulation and Training Technology Center in Orlando, Florida and is responsible for the research and development of applied simulation technologies for learning, training, testing and mission rehearsal.

Andrew Bertrand is a Staff Software Engineer at the Virtual Heroes Division of Applied Research Associates, specializing in GUI and API development.

Haris Javaid is a Junior Software Engineer at the Virtual Heroes Division of Applied Research Associates, specializing in mobile and web services development.

Michael Shin is a Staff Software Engineer at the Virtual Heroes Division of Applied Research Associates, specializing in C++ development and computational geometry.

Measuring Rapid Streaming Delivery of Synthetic Environments

Greg Peele, Andrew Bertrand, Haris Javaid, Michael Shin
Applied Research Associates, Inc.
Orlando, FL
gpeele@ara.com, abertrand@ara.com, hjavaid@ara.com, mshin@ara.com

Julio de la Cruz
ARL HRED ATSD
Orlando, FL
julio.a.delacruz4.civ@mail.mil

BACKGROUND AND MOTIVATION

Synthetic environments provide critical data for a wide range of models, simulations, and operational applications. They are the active backdrop upon which simulation and game platforms rely for modeling physics effects and influencing autonomous behaviors at varying levels of resolution. In addition, they can serve as navigational and planning aides for manned (e.g., operational tactical maps), semi-autonomous, and autonomous activities (Menozzi et al, 2015). A synthetic environment must provide a way to represent the shape and characteristics of its components at the fidelity required by the runtime software it supports.

The on-disk and in-memory representation of military synthetic environments have traditionally been tied to implementation-specific formats. Each of these implementations was narrowly tailored to meet the simulation's requirements, taking into account not only the limited set of geometry types and attribution sets needed to fulfill the immediate needs of the system, but also the hardware constraints (e.g., available RAM, CPU architecture, etc.) imposed by the simulation platform. These design choices complicate adapting these synthetic environments to support new content and new platforms.

The rise in high-resolution data availability and its relevance for use in simulated training and operational environments have highlighted the need for new approaches for synthetic environment storage and distribution (Peele, Adkison, de la Cruz, & Borkman, 2011). Currently, the Open Geospatial Consortium (OGC) provides web standards for distributing vector and raster data. The reference implementation for these standards is the open-source GeoServer spatial data server software. GeoServer's genesis is heavily centered on serving the needs of the cartographic community. However, these needs are evolving with the ubiquity of mobile and cloud-based computing technologies.

Much research is being done on the use of mobile and cloud computing technologies as complimentary simulation platforms (e.g., Wen et al., 2013; Sarmiento, Amor, Padron, & Regueiro, 2011; Sarmiento et al., 2012, Santoso, 2014). Significant issues remain to be solved: Synthetic environments are big data sources, a fact underscored in Table 3 of the RESULTS section of this paper. Date (2016) mentions that it is often more practical to ship big data sources than send them over the network. Fernando et al. (2013) highlights low bandwidth, mobility, and limited storage as challenges in mobile computing, especially regarding the I/O cost of retrieving and storing data from the cloud.

Current OGC standards, which are XML-based, are verbose and thus perform poorly when used on mobile and cloud-based platforms. This makes them unsuitable for synthetic environment applications that require frequent updates. Deploying synthetic environment data on mobile and cloud-based platforms necessitates reducing the encoding, transmission, decoding, and storage cost of synthetic environments in alternate formats with respect to conventional web standards. Needed are standardized and network-streamable ways to retrieve and update synthetic environment content filtered by context and location. Transmission and storage formats should be compact and easily interpreted to reduce the overhead of network transmission, data conversion, and data storage on resource-constrained environments, whether such limitations arise from lack of storage space, network accessibility, computational capabilities, or a combination of these factors.

EXPERIMENT DESIGN

Metrics and Rationale

This paper establishes a constrained and easily reproducible testing methodology to measure bandwidth, storage, and latency of synthetic environment content, specifically assessing systems for use cases involving mobile devices and cloud computing. We apply our methodology to several OGC XML streaming standards to serve as the baseline for future evaluations to accurately quantify the benefit of newer approaches. The current version of OGC's official test suite (OGC Testbed 11) was tailored for interoperability and standards-compliance testing (Open Geospatial Consortium, 2014), whereas the proposed methodology focuses on performance metrics. The upcoming OGC Testbed 12 plans to include a Compression and Generalization area (Open Geospatial Consortium, 2015) that may be more compatible with the metrics we propose in this paper.

The below paragraphs discuss details about each test use case, as well as metrics suited for these use cases.

1) **Mobile devices in network-denied or network-constrained environments**
   Mobile devices offer the portability and convenience of accessing environment data in the field. However, in field environments, network streaming may be unavailable, slow, or unreliable. Therefore, in network-constrained or network-denied environments, data should be stored on the mobile device. Since mobile devices are often limited in hard disk capacity, database files should be small. When network speed is slow, requests must be efficient in order to scale with the reduced throughput and capacity of the network. Because mobile devices are often limited in power and memory, decoding of the data retrieved from the server must be efficient.

2) **Cloud-computing environments**
   In cloud computing environments, servers may handle many streaming requests at once, and data may be distributed across many nodes. Therefore, requests must be processed quickly and efficiently. Beyond file transfer speed, encoding and decoding times of transfer formats must be considered.

From these considerations, we identify several metrics that determine how well the system will perform in each use case:

1) **On-disk file size**
   When network access is limited, data cannot be streamed in piecemeal by pages. Rather, during infrequent windows of network access, data must be downloaded in full to disk. Because mobile devices have limited disk space, it is vital to limit the file size of data retrieved from the server. Detailed feature and elevation data require a lot of storage, especially if they cover a large area. A central server that handles various requests from many clients could potentially have much data. Therefore, it is always beneficial to limit the size of the data stored on the server.

2) **Transfer file size**
   In a constrained network, only small amounts of data may be forced through over a reasonable duration, so it will be advantageous if transfer files are smaller. A high-capacity network enjoys similar benefits if there is high traffic. When there are many clients trying to access a server on the same network, throughput may be limited, and reductions in the transfer file size would enable proportionately more clients to access the server at the same time. Small transfer file sizes result in a naturally faster file transfer speed.

3) **Latency of requests**
   Low latency reduces overall file transfer speed. In network-constrained situations, it may not be clear whether or not the request was successfully received by the server. Low latency allows faster verification of a successful connection, which is valuable in time-sensitive scenarios. For a server handling a large volume of requests, low latency is vital for avoiding collisions and bottlenecks of simultaneous or near-simultaneous requests.

4) **File transfer speed**
   High file transfer speed is beneficial for the same reasons that small transfer file sizes are beneficial to network-constrained or high-volume request use cases. High file transfer speed reduces the length of the connection necessary between the server and the client. In a constrained network, connection may be intermittent and

unpredictable, so a faster data transfer increases the likelihood of a successful transfer. For high-volume servers, high file transfer speed improves throughput.

5) **File transfer peak bandwidth use**
File transfer peak bandwidth use indicates how efficiently the process can make use of the physical throughput. The service's efficiency should scale directly with improvements to the architecture.

6) **Memory use**
Memory use should remain low for decoding and encoding on both clients and servers. Mobile devices have limited memory, and a server runs the risk of running out of memory if it handles many requests.

7) **Decoding algorithm speed**
The software algorithm for encoding and decoding data from the network transfer protocol can take up a significant amount of the total time spent from when the request is initiated until usable data is received. Transfer protocols are typically optimized for readability or minimum size and must be decoded into a form usable by the final software application. Fast and efficient decoding is especially important on mobile devices and servers handling many simultaneous requests, because computing power is limited.

For our data sets, we obtained raw Virtual Globe terrain data for the entire continental US from the Defense Threat Reduction Agency. We hosted this data on a GeoServer[1] instance.

The data contains shapefiles and GeoTIFF files representing elevation. The data is organized in a geotile-aligned structure similar to a CDB, but it only has a single level of detail. Figure 1 outlines the geographic regions. Table 1 outlines each data set's longitudinal and latitudinal extents; each data set covers a 2 x 2 geotile grid. All data is accessible via the Web Feature Service or WFS (Vretanos, 2010) and the Web Coverage Service or WCS (Baumann, 2012): GeoServer uses WFS as a protocol for sharing and manipulating spatial vector data over the web, using the XML-based Geographic Markup Language (GML) as the interchange format. WCS enables GeoServer to provide a way to retrieve and manipulate raster data, using GeoTIFF as the default file format. Finally, GeoServer uses the Web Mapping Service (WMS) to create and retrieve map images.

Figure 1. A globe view of the geotiles represented by the data sets.

[1] http://geoserver.org/

Table 1. The longitudinal and latitudinal extents of each data set in degrees.

Point of Interest	West	East	South	North
Albuquerque, NM	107° W	105° W	34° N	36° N
New York City, NY	75° W	73° W	40° N	42° N
Yellowstone Park, WY	112° W	110° W	44° N	46° N
Raleigh, NC	80° W	78° W	35° N	37° N
New Orleans, LA	91° W	89° W	29° N	31° N

Methodology

We tested the performance of the OGC WFS and WCS XML-based approaches on different data types in multiple geographic regions. Our methodology enables us to later make direct comparisons to various other repository and transmission protocol approaches, targeting specific aspects of the data retrieval process in order to make accurate comparisons based on the specified criteria.

To retrieve terrain data, we set up our own lightweight HTTP client in C++. The client used the open-source library libcurl [2] to send and receive hard-coded HTTP requests. Using POST and GET requests, we retrieved data using WFS and WCS standards. This allowed us to easily test specific aspects of performance and separate different steps of the retrieval process.

We measured the response time of WFS and WCS queries using the Wireshark[3] tool. Using our HTTP client, we ran queries to obtain the data of each layer in standard GML format. The queries included requesting each data layer separately, as well as requesting all layers in a region at once. We measured the total time needed to receive the full data, along with the latency of the response and the peak bandwidth of the transfer (latency, for our purposes, is the time between sending the first query and receiving the first response. This includes both network latency and time for encoding the data into the GML format). We accounted for network handshaking protocols, in that we removed handshaking times from the overall time. We also used Wireshark to measure the total amount of data transferred in each query (we removed the handshaking protocols from the overall data transfer size). We also tracked the server-side raw data file sizes and the client-side GML file sizes.

The JProfiler[4] tool allowed us to track memory usage of GeoServer during retrieval requests. We tracked both the highest and lowest memory usage. We used the Intel Inspector[5] tool to profile client-side memory usage, similarly tracking the highest and lowest memory usage.

Finally, we tested the time needed for a minimum processing of GML-formatted data, with a simple libxml2[6]-based parser. The parser iterated through XML elements determining the minimum time needed to process the XML-based data. We tested data retrieved via both WFS and WCS; to do so, we used both SAX and DOM-based approaches, in order to make a full comparison.

This methodology lays the groundwork for future analysis and comparisons of upcoming repository and protocol standards and approaches. Several ongoing development and standardization efforts aim toward comprehensive synthetic environments designed for simulation and runtime performance requirements. For example, the OGC CDB draft standard (Reed, 2016) defines a binary file-based repository for a multi-resolution runtime environment. The Khronos Group GL Transmission Format or glTF (Khronos Group, 2016) optimizes binary transmission of streamlined visual content. The Army Research Lab's Layered Terrain Format or LTF (Peele, Adkison, de la Cruz, & Borkman, 2011; Borkman, Peele, & Campbell, 2007) provides extensible compressed streaming of a comprehensive environment protocol backed by a file cache. Researchers can easily target different phases of the

[2] https://curl.haxx.se/libcurl/
[3] https://www.wireshark.org/
[4] https://www.ej-technologies.com/products/jprofiler/overview.html
[5] https://software.intel.com/en-us/intel-inspector-xe
[6] http://xmlsoft.org/

data request process in order to compare the efficiency of current XML-based approaches current XML-based approaches to the aforementioned upcoming approaches.

Hardware

We tested the OGC GeoServer services using two Alienware Aurora R4 desktop machines. The systems' specifications are shown in Table 2.

Table 2. Client and server hardware information.

	Server	Client
OS:	Windows 7 Professional 64-bit	Windows 7 Professional 64-bit
Processor:	Intel Core i7-3930K – 3.8GHz – 6 Core	Intel Core i7-4820K – 3.70GHz – 4 Core
Storage:	Western Digital Black 1TB 7200RPM	Dell LiteOn 256GB SSD
Memory:	1600MHz 16GB DDR3	1600MHz 16GB DDR3
Graphics:	NVIDIA GeForce GTX 555	NVIDIA GeForce GTX 760
Software:	GeoServer 2.9 JProfiler	Wireshark Intel Inspector C++ HTTP Client (cURL)

RESULTS

We compared the original source data with the OGC protocol counterparts. In all cases, the equivalent OGC data required more bytes than its source format. The feature- and attribution-intensive GML files were more than twice the size of the original ESRI shape file data sets [the geometry (shp), the index (shx), and the attribution table (dbf)] due to the verbose nature of XML encoding. This brings into question the practicality of subjecting large data sets to this kind of XML encoding. The WCS data was not significantly larger than the original Digital Elevation Model (DEM) files, because both formats consist of raster binary data. During the transfers for data retrieval, the peak memory of the GeoServer was determined to be 1,912 MB using the JProfiler application.

Table 3. GeoServer data set original and transfer file sizes

Data Set	Service	Server File Size (megabytes)	Transfer Size (megabytes)
Albuquerque	WFS	872.50	2,214.43
	WCS	2,018,92	2,110.10
New Orleans	WFS	2,853.70	3,277.95
	WCS	2,018.92	2,057.27
New York	WFS	5,171.82	13,876.38
	WCS	2,018.92	2,055.28
Raleigh	WFS	2,128.57	4,471.60
	WCS	2,018.92	2,084.78
Yellowstone Park	WFS	256.74	669.84
	WCS	2,018.92	2,089.61

A key metric we tracked is the transfer rate for data retrieval. The WFS data was able to hit a peak transfer rate close to 100 megabits per second; however, the average rate was significantly lower. This suggests the server had to spend a significant amount of time retrieving the data from disk and encoding it into XML. Future work to instrument GeoServer to break down this metric into individual components may identify specific bottlenecks. The initial latency from request to response was consistently small (consistently less than one second) as the server always responded immediately with header data before streaming the full data set results.

Table 4. GeoServer data set transfer metrics.

Data Set	Service	Latency (milliseconds)	Duration (seconds)	Bandwidth (megabits per second)	
				Average	Peak
Albuquerque	WFS	84.51	190.16	61.29	96.66
	WCS	207.07	307.95	55.18	75.97
New Orleans	WFS	207.28	421.87	60.14	96.38
	WCS	209.95	266.33	61.80	77.39
New York	WFS	99.46	1283.67	74.00	96.83
	WCS	22.10	264.77	62.10	77.98
Raleigh	WFS	212.87	396.71	62.71	96.48
	WCS	218.61	263.50	63.30	77.00
Yellowstone Park	WFS	96.88	58.55	55.63	96.44
	WCS	221.70	265.85	62.92	77.62

The final major metric we tracked is the time it takes to process received GeoServer layers. For the GML case, we were able to set up a streaming parser and DOM parser using the libxml2 open-source library. We only benchmarked the time it took to iterate through all of the element nodes without further processing. Streaming was the more time-efficient technique, because it made no impact on the memory consumption of the application. The DOM parsing was magnitudes slower, because the XML had to be allocated in memory. More memory consumption can be expected in typical usage where the data is actually manipulated. When parsing New York and Raleigh data, the client machine ran out of memory and began paging to the disk. Since the client had a solid-state drive, the impact of paging was not as drastic as it would be for a conventional hard drive. WCS data was parsed using GDAL 1.11[7], an industry-wide used library for handling raster format files.

Table 5. GeoServer data XML parsing time.

Data Set	Service	Streaming Time (seconds)	DOM Parse Time (seconds)	Peak Memory (megabytes)
Albuquerque	WFS	52.95	43.45	4,704
	WCS		24.76	2,120
New Orleans	WFS	75.23	679.24	14,043
	WCS		23.35	2,068
New York	WFS	412.78	7427.34	16,384
	WCS		22.74	2,066
Raleigh	WFS	118.15	2325.50	16,384
	WCS		22.57	2,099
Yellowstone Park	WFS	13.43	27.37	2,186
	WCS		22.57	2,103

CONCLUSIONS

This experiment was extremely useful in identifying several important performance characteristics of the OGC XML web services. Some key takeaways observed in the metrics are as follows:

- The network transfers were unable to fully saturate a 100 megabits per second network link
- GML encoding is significantly less space-efficient than original shape file encoding
- DOM parsing of large GML data sets is extremely impractical, even on high-end desktop machines
- XML decoding time is consistently at least 25% of total transfer time, even with streaming parsing

Based on the metrics, it appears that scaling down to slower networks such as 54 megabits per second (Mbps) 802.11g or 40 Mbps 4G LTE would somewhat slow data streaming performance. Scaling down further to 11 Mbps

[7] http://www.gdal.org/

802.11b or 7.2 Mbps 3G GSM would significantly compromise performance. Conversely, increasing the speed to 1 gigabit per second (Gbps) may not provide much benefit, as evidenced by the fact that the current experiment was unable to consistently saturate the 100 Mbps link. In particular, New Orleans and Raleigh data sets featured significant portions where bandwidth usage was unexpectedly low, as shown in Figure 2.

It is noteworthy that neither the repository format nor WFS and WCS transmission protocols used any form of compression. We expect other standards and approaches that use compression techniques would show significant reductions in transfer time, at the cost of more CPU time spent encoding and decoding the data. Whether the data reduction would offset the CPU encode/decode time is an open question.

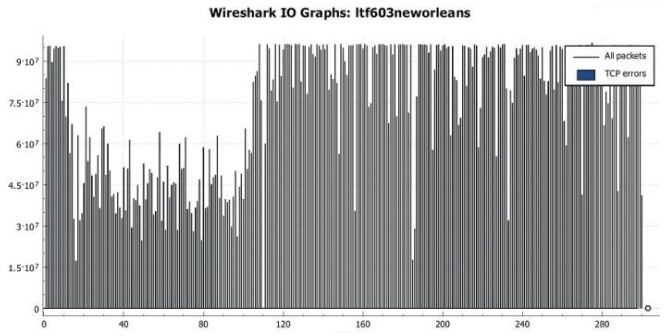

Figure 2. New Orleans data Wireshark transfer speeds.

The total size of the retrieved data sets was just above 30 gigabytes. It is important to note that GML and even conventional source data file sizes are far too large for flash storage devices commonly used in mobile devices and cloud computing infrastructures. However, these files are still workable with conventional hard disk storage. Unfortunately, large file sizes create the risk of slower transfer rates due to bottlenecks from hard drive read and write speeds.

Possible performance metrics for future testing include GML data encoding time on the GeoServer application and the performance difference between the Java-based GeoServer and a native C++ application that presents the same operations and capabilities.

FUTURE WORK

This experiment methodology lays the groundwork to conduct a series of future experiments analyzing the improvements of new and upcoming standards, repository types, and streaming protocols. The OGC CDB and GeoPackage formats are ideal candidates for the repository side of testing, to ensure they match or exceed the performance and disk efficiency of conventional source data. The performance comparison between these two is also of interest for areas in which their use cases overlap.

On the streaming side, further development of LTF relies on characterizing and quantifying performance gains of its unified object model and Google Protocol Buffer encoding over conventional disjoint XML approaches. For streaming 2D and 3D visualization, measuring the latency from initial data request to first-rendered GPU frame is a potentially very useful metric. It would quantify the benefit of the specialized glTF protocol over more general protocols such as CityGML or LTF and historical model formats such as OpenFlight. This performance analysis would help guide service developers to determine whether there is sufficient benefit to warrant including glTF as a streaming option for their clients.

The experiment methodology itself should be extended and refined in collaboration with OGC. The data sets used in our experiment unfortunately have distribution restrictions; the better solution will be to identify and provide a standard open reference data set for each type of environment region. In addition, our experiment solely tested the batch transfer use case, in which large amounts of data are transferred at once. Another important usage pattern is tiled access of small adjacent regions in sequence, such as would be done by a simulator or game engine. Overall, having a common experiment data set, methodology, and control group metrics will aid governments, standards organizations, and potential product buyers to objectively analyze the strengths, weaknesses, and use cases for the range of available synthetic environment formats and protocols.

ACKNOWLEDGEMENTS

We would like to thank US Army Research Lab (ARL) Human Research and Engineering Directorate (HRED) Advanced Training and Simulation Division (ATSD) for sponsoring this paper and experiment as part of the development of the Layered Terrain Format specification. We would also like to thank the Open Geospatial Consortium 3D Information Modeling (3DIM) Domain Working Group for their helpful input and advice.

REFERENCES

Baumann, P. (2012). OGC WCS 2.0 Interface Standard – Core: Corrigendum. Open Geospatial Consortium Specification, OGC 09-110r4.

Borkman, S., Peele, G., & Campbell, C. (2007). An Optimized Synthetic Environment Representation Developed for OneTESS Live Training. In Interservice/Industry Training, Simulation, and Education Conference.

Date, S. (2016). Should You Upload or Ship Big Data to the Cloud? In Queue – Cloud Debugging. Volume 14 Issue 2, March-April 2016. p. 30. ACM New York, NY, USA.

Fernando, N., Loke, S., & Rahayu, W (2013). Mobile cloud computing: A survey. Future Generation Computer Systems 29.1 pp 84-106.

Khronos Group (2016). The GL Transmission Format (glTF) Specification Version 1.0.

Menozzi, A., Hansen, J., Peele, G., Snarski, S., & Merrick, C. (2015). On the development of a framework for underwater localization using all-source data. In OCEANS 2015-Genova (pp. 1-10). IEEE.

Open Geospatial Consortium. (2014). Request for Quotations (RFQ) and Call for Participation (CFP) for OGC Testbed 11 – Annex B: OGC Testbed 11 Architecture. Retrieved June 1, 2016 from https://portal.opengeospatial.org/files/?artifact_id=60889.

Open Geospatial Consortium. (2015). Request for Quotations (RFQ) and Call for Participation (CFP) for OGC Testbed 12 – Annex A: Testbed 12 Management Requirements. Retrieved June 1, 2016 from https://portal.opengeospatial.org/files/?artifact_id=65483.

Peele, G., Adkison, J., de la Cruz, J., & Borkman, S. (2011). Building a compact high-fidelity runtime database engine layer by layer. Proceedings of IMAGE Conference.

Reed, C. (2016). Volume 1: OGC CDB Core Standard Model and Physical Data Store Structure. Open Geospatial Consortium Candidate Standard, OGC 15-113.

Santoso, G. A. R. (2014). GPS-Based AR Games Development Potential. SISFORMA, 1(2), 5-8.

Sarmiento, A., Amor, M., Padron, E., & Regueiro, C. (2011). An analysis of android smartphones as a platform for augmented reality games. Proceedings of UBICOMM.

Sarmiento, A. L., Amor, M., Padrón, E. J., Regueiro, C. V., Concheiro, R., & Quintıa, P. (2012). Evaluating performance of android systems as a platform for augmented reality applications. International Journal on Advances in Software Volume 5, Number 3 & 4, 2012.

Vretanos, P. A. (2010). OpenGIS Web Feature Service 2.0 Interface Standard. Open Geospatial Consortium Implementation Standard, OGC 09-025r1 and ISO/DIS 19142.

Wen, Y., Chen, M., Lu, G., Lin, H., He, L., & Yue, S. (2013). Prototyping an open environment for sharing geographical analysis models on cloud computing platform. International Journal of Digital Earth, 6(4), 356-382.