2009

Efficient XML Interchange: Compact, Efficient, and Standards-Based XML

Snyder, Sheldon

http://hdl.handle.net/10945/30774
Efficient XML Interchange: Compact, Efficient, and Standards-Based XML for Modeling and Simulation

Sheldon Snyder
Don McGregor
Don Brutzman
Naval Postgraduate School
700 Dyer Rd
Monterey, CA 93940
831-656-7605, 831-656-2149
slnsnyder@navy.mil, mcgredo@nps.edu, brutzman@nps.navy.mil

Keywords:
EXI, XML, Compression, Optimized XML, Efficient XML, Binary XML

ABSTRACT: XML has become a popular representation format for data, both in modeling and simulation and elsewhere. However, XML’s design choice of a text-based format also makes XML data files much larger than binary files, making XML languages difficult to use in bandwidth-constrained military applications. This limitation has resulted in several ad-hoc attempts to make XML more compact, each of which tends to be incompatible with the other. Efficient XML Interchange (EXI) is a World Wide Web Consortium (W3C) Working Draft for the compact and efficient representation of the XML infoset. EXI is designed to be generally applicable to all XML documents, and lays the foundation for a unified format for compact XML document representation.

This paper presents compactness results for several popular modeling and simulation XML file formats, including Distributed Interactive Simulation (DIS), Scalable Vector Graphics (SVG) and Extensible 3D Graphics (X3D). Recent commercial and open source EXI implementations are also described.

1. Rationale for Compact XML

1.1 Introduction

Extensible Markup Language (XML) has become one of the most widespread data representation formats in Department of Defense (DoD) Modeling and Simulation (M&S) applications. XML’s success in the internet generally, and M&S in particular has largely flowed from critical design decisions made by the World Wide Web Consortium (W3C) regarding what XML is and is not. Many of these decisions are mentioned in the W3C’s XML in Ten Points document from 1999 [1].

In an M&S context the most critical features mentioned by this document include

- XML is for structuring data
- XML is text, but isn’t meant to be read
- XML is verbose by design
- XML is a family of technologies
- XML is modular
- XML is license-free, platform-independent, and well-supported.

M&S exploits XML to describe and structure data in a modular way, and can use the full range of XML technologies such as parsers, web services, XSLT, digital signatures, and other tools. Its platform independence is an absolute requirement in the heterogeneous DoD environment.

But the two other points—“XML is text”, and “XML is verbose by design”—are more problematic. M&S data is often very large and heavily numeric. Representing the data in text format can dramatically increase the size of datasets. Representing the data as text also increases the load on CPUs when the numeric data in text format is translated to binary form in order to perform calculations. Verbosity detracts from the ability to send data across the network. Practitioners put up with these aspects of XML in order to gain interoperability and a standard method of representing data. This has on the whole been very successful; to the extent there is a standard for structuring data, it is XML. The widespread use of XML has led to network
effects: the more people using XML, the more useful XML becomes. The very ubiquity of XML has become one of its major benefits. Within the well-connected Global Information Grid (GIG), or when using servers and clients with grid-supplied electrical power and wired connections, XML has become a viable and useful format.

However, XML architectures can exclude edge devices on the GIG, particularly in military environments. The military often operates in environments with limited bandwidth. Handheld devices are battery-powered, have less powerful CPUs, and can’t process as much XML as desktop or server devices. The benefits gained by adopting XML in the server environment can exclude devices at the muddy, boots-on-the-ground edge of the network and prevent a grand unification of all data formats. Even within a server room environment there is benefit to having compact and efficient representations of XML when sending it across the network or when high performance parsing is required.

What is needed is a compact representation of XML that gives up the “text based” “verbose by design”, and “human readable” design goals of XML in order to achieve compactness and processing efficiency, while retaining compatibility and interoperability with other XML tools. By relaxing the text-based design restrictions, XML can be extended to edge devices where XML would not otherwise be viable. It should also be efficient to parse and standards-based. Crucially, there should be a single standard adopted throughout DoD for compact and efficient XML representations.

1.2 Compact XML Representations

The primary rationale for a non-text XML infoset encoding is to grow the web to serve otherwise unsupported use cases. That is, cases that without a compact XML representation cannot use XML. The goal is to make XML technology useful in environments that were previously unable to take advantage of XML, including mobile devices and environments with limited bandwidth. [2].

The W3C conducted analysis to define the requirement gaps between XML and uses cases that were not able to use XML. They devised a comprehensive list of 18 use case domains that were found to be unable to adequately or completely support XML utilization due to size, processing complexity, and memory footprint[2][3].

Many organizations have been aware of XML’s limits and have developed XML compression techniques to overcome the limitations of XML for their problem domain. However, these formats often lack standards organization support, and many of the techniques are not fully adaptable to the entire XML “stack” of tools and technologies. To mention one example, it should be possible to encode an XML document in a more compact representation and also retain the ability to confirm a digital signature of the document when it is converted back to XML.

Efficient XML Interchange

Efficient XML Interchange (EXI) is a W3C standards effort to create a compact and efficient representation of the XML infoset [4]. It is currently in “Last Call” status in the W3C standards process, which is a period of public comment on the proposed standard. It is intended to meet all the requirements discussed above. It is able to represent any XML document— with schema or without schema—and is able to be integrated with the existing XML stack. Since it encodes the XML infoset, any EXI document can be converted back to an XML document.

EXI employs a finite state machine learning techniques to discover the structure of the XML document, which uses along with redundancy checks to generate compact identifier tokens to represent the XML content [5]. This can be accomplished through the pre-parsing of a supporting schema or it can be learned for schemaless documents during the parsing of the XML document itself.

EXI is designed to efficiently represent all XML documents while reducing bandwidth, processing workload and memory requirements, preserves battery life on mobile devices, and supports all of the features specified as the minimum necessary by the W3C’s XML Binary Characterization Working Group [6][7].

GZip

GNU zip (GZIP) and similar technologies (zip and zlib) are the most common desktop compression tools. They use variants of the Lempel-Ziv 1977 (LZ77) algorithm to perform compression [8]. The key to the compression algorithm is duplicate string representation. The second and successive occurrences of a string are replaced by a compact pointer to the first occurrences in the form of a paired list of the distance (up to 32K) and length (258 bytes). If the second and follow on occurrences fall outside of this range, it is not referenced and instead written as a literal (1:1 sequence of bytes). Huffman trees are the basis of the compression. Once a block size of data has been stored in the tree, it is deflated and written to file.
Android  Android, a mobile device platform from Google, uses a binary XML format [9]. Android’s binary format was developed for many of the same reasons as EXI: it is designed for a mobile device environment that requires low power, has limited memory and the limited bandwidth provided by a cell phone link. There has apparently been no effort to submit the Android binary XML format to a standards organization.

ASN.1  The Abstract Syntax Notation 1 (ASN.1) is a International Telecommunications Union (ITU) standard for describing data structures, and for encoding and decoding those data structures in a variety of formats. One of these formats is XER, XML Encoding Rules. ASN.1-defined structures can be encoded in other formats, including Distinguished Encoding Rules (DER) and Packed Encoding Rules (PER). While ASN.1 does not directly address the encoding of XML documents, it is possible to process an existing XML document into one of the ASN.1 encoding formats.

Extensible Schema-Based Compression  Extensible Schema-Based Compression (XSBC) encodes a XML document based on a schema into a binary format that is more compact and faster to parse than textual XML [10]. Based on the schema, numeric tags are used to replace XML tags, and schema information is used to save text numeric data in binary format. The main advantage XSBC is its simplicity. The main disadvantage is it requires a schema.

Fast Infoset  Fast Infoset is an open, standards-based binary format based on the XML Information Set [7]. Like the other candidates, the use of tables and indexing is the primary mechanism for compression. It is possible via the use of encoding algorithms to selectively apply redundancy-based compression or optimized encodings to certain fragments. Using this capability, as well as other advanced features, it is possible to tune the "sweet spot" for a particular application domain.

Need For A Single Interoperable Standard  To a large degree XML’s success is due to its interoperability and standards compliance, and a single standardized XML binary format is needed to ensure continued universal interoperability. A standard XML binary format prevents multiple splinter binary formats from developing, which if not discouraged will reduce the interoperability of XML, its cornerstone of success [11][5].

1.4 W3C Efficient XML Interchange

Efficient XML Interchange (EXI) is the W3C’s standards-based format for the compact and efficient representation of XML. It is based on the earlier work from a commercial implementation by the company Agile Delta [15]. EXI is a general purpose format that has shown to work on the entire range of the XML family of languages [5]. The EXI format is very adaptive and flexible and achieves compactness results equal to, and normally superior to, alternative formats tested. EXI supports the leveraging of the XML architecture with a number of XML specific encoding options and extensions that deliver superior performance gains for all domain cases [7][6]. A brief description of EXI follows.

String Tables  In general, the strategy of EXI is to replace repeating string tokens with shorter binary compact identifiers, and in the case of schema-informed documents to write numeric values in binary form. String tables are the backbone of the compact identifier assignment process. Each content item (string event) is recorded in a string table, with one table for each XML namespace. The first time a string value is encountered, its literal string is written to the EXI stream directly as a sequence of bytes, and then added to the string table. If seen again, this repeated occurrence is replaced with a compact identifier (index into a string table) instead of the string literal. This process delivers a file savings even if a XML document contains a large number of unique non-repeating string values. In the worst case, every string in the XML document is unique, a savings is still achieved by not writing the {=, ', "", <, >, </} XML formatting characters.

Grammars and Events  XML documents as a whole cannot be described by a single Chomsky regular language or a single finite state automata. Instead, EXI uses a stack of finite state automatons, one for each portion of the document that can be described as a regular language. EXI Grammars (which are equivalent to an automata) define the structure of an XML document, defining which EXI events (similar to XML SAX events) can occur, when they can occur, and how many have occurred [5]. Both EXI grammars and EXI events utilize a growing (learning) token to represent their values within an EXI stream.

A grammar represents a level of the XML document hierarchy. A simple XML fragment is shown in listing 1. This fragment can be described in five grammars: document, notebook, note, subject, body. Note that all
XML documents start with the document grammar by default.

```xml
<notebook date="2007-09-12">
  <note date="2007-07-23" category="EXI">
    <subject>EXI</subject>
    <body>Do not forget it!</body>
  </note>
</notebook>
```

**Listing 1. XML Notebook Example**

EXI events follow the standard XML parsing events such as startElement, Attribute, ProcessingInstruction and others. Events are coded by a sequence of one to three non-negative integers called parts that uniquely identify an event based on the current grammar rule set. The grammar rule sets are defined so that the smallest number of bits needed to represent all possible events within a grammar are used based on the current grammar event count, and the likelihood of the event’s occurrence in a typical XML document. For example, an Element-Grammar events are encoded using the following rules show in Listing 2.

```xml
ElementContent:
  EE 0
  ChildContentItems (1.0)
ChildContentItems (n.m):
  SE (*) ElementContent n.m
  CH ElementContent n.(m+1)
  ER ElementContent n.(m+2)
  CM ElementContent n.(m+3).0
  PI ElementContent n.(m+3).1
```

**Listing 2. An XML Element Grammar**

Here, “n” is the number of events currently contained in the grammar (part 1), “m” is a 2\textsuperscript{nd} level unique event id (part 2), and the 0 or 1 are the 3\textsuperscript{rd} level relatively rare event ids (part 3).

The fewest number of bits needed to encode any of the parts of an event or compact identifier are found by taking the ceiling \( \log_2 s = c \) where \( s \) is the number of distinct entries (grammar event count or string table size) and \( c \) is the number of bits.

**Output Formats** EXI’s compact identifiers can be of variable length; for example, a compact identifier “1” can be represented in a single bit when written to a file. However, one of the options for EXI is to write compact identifiers on byte boundaries. For schema-informed documents, element attributes can be identified as numeric values. Numeric values can also be written as variable-length data or on byte boundaries. Floating point values are represented in an internal, variable length format rather than IEEE-754 format. This was done because text XML documents can represent values not expressible in IEEE-754 format. Using IEEE-754 format to encode floating point values would therefore make EXI an inherently lossy compression scheme. EXI allows pluggable codecs that can encode values in native IEEE-754 format.

**Coding Options** EXI support several XML pruning options that permit the EXI processor to bypass certain events: comments, processing instructions, DTD and Entity declarations, and namespaces. Often these events do little for the applications that read XML documents and can be disregarded [5]. Comments for example are purely for human readability and add to the file size. At the other extreme, EXI can preserve all information, such as white space, in order to maintain round-trip XML->EXI->XML digital signature compatibility.

EXI also offers coding options to handle pathological cases that may adversely affect performance, such as documents that cause very large string table sizes, documents that deviate from the XML schema that describe them, and other cases.

2 Application of EXI to DoD M&S XML Documents

2.1 DoD is Heavily Investing in XML

Within DoD, the Net-Centric and Force-Net visions of a system of systems is to seamlessly integrate data from numerous sources into a common picture that tacticians can pull information as desired, and do so within a meaningful, and relevant context of the mission. In support of this vision, DoD has mandated the use of XML in all new systems, and the push to back fit XML into the existing stovepipe systems to ensure all systems interoperate [12][13].

2.2 DoD Files are Numerically Intensive

DoD tactical data links exchange data in terms of quantifiable parameters [12]: degrees, percentages, counts, probabilities, trajectories, and other numerically based data. These data links are the communication medium between machines, and are not intended to be directly readable by humans. They live to aim weapons, direct troops, move ships, relay operational status reports of deployed units, and so do not require descriptive string values as humans are not directly reading the message formats.
For a case example, an Air Tasking Order is transmitted daily listing all air traffic operations. This is a very large file (50MB+) containing flight number, weapons configurations, refuel locations, launch/recover times, and a continuing list of numerical parameters that define every flight for that day. Without a tool to parse and label the ATO file data, the data would be meaningless to humans.

2.3 M&S Files are Large

When military units are not engaged in operational activities they train in simulators. These simulators enable DoD forces to exercise in any theater of battle around the world as long as the force Orders of Battles (OOB) and landscape are defined. The simulation file(s) that define the OOBs and landscape can be tens to hundreds of megabyte in size as they must list every possible person, bullet, tank, gallon of gas, and other minute details to deliver the required fidelity to effectively exercise combat units in a simulation environment. In addition to being very large descriptive files, they are almost entirely numerical values.

2.4 Next Generation of Devices

The next generation of DoD networking will likely involve mobile handheld devices. Text-based XML formats are difficult to process on these devices because of limited CPU power, battery size, and bandwidth capability. A more compact and efficient XML format will enable XML to extend to the next generation of war fighting IT devices as if they were traditional desktop PC.

Given IPv6 enables the placement of an IP address every few centimeters across the world and the proliferation of inexpensive portable devices the importance of mobile devices will grow. The next generation devices must process XML as the GIG is XML.

2.5 DoD Tactical Networks Bandwidth

Battlefield bandwidth is always limited; there are no wires running into battle to keep the forces connected. On naval vessels hull mounted satellite connections are all that is available. Since these hull-mounted antennas are in constant motion, connectivity is usually degraded.

For example, deployed US Navy commands are often limited to a level of service equivalent to that of a dial-up modem or less to provide all networking needs of the command. These commands must operate under these conditions to exchange Tomahawk orders, transfer intelligence imagery, as well as let its sailors send email home to mom. As file sizes grow, the capability of the command’s network diminishes as it must service a single need. An 85MB high resolution satellite intelligence imagery file would take approximately 3.5 hours to transfer (figure 1), and that is assuming the full dedication of the entire network’s resources, and a constant clean connection. Neither assumption is representative of reality, even under the best conditions.

3. Experimental Results

Some of the following use cases are specific to DoD M&S, but also include other XML languages in order to demonstrate EXI’s flexibility.

3.1 Results Comparison

The following results were collected using the Siemens open source EXI engine [16] using the default preservation options, and with the extra compression encoding option.

3.2 XML Language Cases

Haar: The OpenCV artificial intelligence vision package uses XML to configure its real time Haar facial detection algorithm [14]. An example of facial recognition is shown in figure 2. Using XML enables developers the ability to rapidly alter the facial detection configuration based on environment and unique features as needed without requiring C code.
Scalable vector graphics (SVG): Is a XML language for defining 2D graphics (figure 3) used for web applications, and is being ported to mobile devices. SVG file from the open sources SVG editor Inkscape [www.inkscape.org].

Figure 3: Example SVG file when rendered

OOB: Military simulations rely upon Orders of Battle (OOB) to define what objects are available in the scenario. XML based OOB enable heterogeneous simulators to use the same battle configuration.

Military Scenario Definition Language (MSDL): Is an XML language that is the standard format for describing the stating state of military action within DoD military simulations.

Autonomous Vehicle Control Language (AVCL): Is an XML languages used in the command and control of autonomous unmanned vehicles to represent mission: planning, scripting, replay.

Discrete Event Simulation: A visual representation of the classic discrete event simulation server queue model is shown in figures 4 and 5. The tool VISKIT [17], based on SIMKIT [18], uses XML as the configuration and storage medium for its models (figures 4 and 5).

Figure 4: Example MM/xQ Event Graph

Figure 5: Example MM/xQ Assembly

X3D: Is a standardized XML language used to define 3D computer graphics (figure 6) that are displayable in web and other browsers.

Figure 6: Example X3D Scene

XHTML: An XML language for representing web content within Web browsers. It is very similar to HTML except that, unlike HTML, XHTML conforms to XML tagging rules.

Humanoid: A Delta3D game engine character mapping that uses XML as the medium to define (figure 7) skeletal structure, texture mapping, motion model, and other human behaviors and attributes.
3D Map: A Delta3D world mapping that uses XML as the medium to define the world (figure 8) terrain, the objects that are in the world, the physics of the world and other attributes.

Notebook: This is the Hello World of EXI, intended to be simple enough to demonstrate the encoding techniques used to create an EXI document from an XML document.

Distributed Interactive Simulation (DIS): DIS is a binary format IEEE standard for M&S. The Open-DIS project [17] can represent the data in DIS in an XML format, and this format can in turn be converted to EXI. The test data included a capture of a number of entity state PDUs.

3.3 EXI Case Results

A compression comparison (table 1) analysis of EXI was conducted against the most common desktop compression tools available in DoD: GZip and Zip. The effects of no operation, GZip, Zip, EXI (schemaless), and EXI (with schema, where available) are listed with resulting file size and percentage of the original input XML file.

<table>
<thead>
<tr>
<th>File</th>
<th>Original</th>
<th>GZip</th>
<th>Zip</th>
<th>EXI</th>
<th>EXI Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAAR</td>
<td>3,748,256</td>
<td>416,948</td>
<td>417,282</td>
<td>286,066</td>
<td>#N/A</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>11.12%</td>
<td>11.13%</td>
<td>7.63%</td>
<td>#N/A</td>
</tr>
<tr>
<td>SVG</td>
<td>4,910</td>
<td>2,238</td>
<td>2,322</td>
<td>2,178</td>
<td>#N/A</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>45.58%</td>
<td>51.57%</td>
<td>44.36%</td>
<td>#N/A</td>
</tr>
<tr>
<td>OOB</td>
<td>3,420,388</td>
<td>194,031</td>
<td>194,325</td>
<td>71,987</td>
<td>#N/A</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>5.67%</td>
<td>5.68%</td>
<td>2.10%</td>
<td>#N/A</td>
</tr>
<tr>
<td>MSDL</td>
<td>3,471,120</td>
<td>262,640</td>
<td>262,940</td>
<td>58,837</td>
<td>#N/A</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>7.57%</td>
<td>7.58%</td>
<td>1.70%</td>
<td>#N/A</td>
</tr>
<tr>
<td>ACVL</td>
<td>10,574,242</td>
<td>754,744</td>
<td>755,048</td>
<td>523,560</td>
<td>444,996</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>7.14%</td>
<td>7.14%</td>
<td>4.95%</td>
<td>4.21%</td>
</tr>
<tr>
<td>DIS Pacs</td>
<td>1,155,742</td>
<td>48,887</td>
<td>49,169</td>
<td>36,180</td>
<td>31,834</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>4.23%</td>
<td>4.25%</td>
<td>3.13%</td>
<td>2.75%</td>
</tr>
<tr>
<td>DIS</td>
<td>136,372</td>
<td>8,545</td>
<td>8,823</td>
<td>6,407</td>
<td>#N/A</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>6.27%</td>
<td>6.47%</td>
<td>4.70%</td>
<td>#N/A</td>
</tr>
<tr>
<td>MMSxQ Graph</td>
<td>3,136</td>
<td>776</td>
<td>1,084</td>
<td>740</td>
<td>447</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>23.34%</td>
<td>32.96%</td>
<td>20.50%</td>
<td>14.86%</td>
</tr>
<tr>
<td>MMSxQ Execute</td>
<td>7,624</td>
<td>1,551</td>
<td>1,863</td>
<td>1,308</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>20.34%</td>
<td>24.44%</td>
<td>24.90%</td>
<td>17.16%</td>
</tr>
<tr>
<td>X3D</td>
<td>7,024</td>
<td>1,000</td>
<td>1,000</td>
<td>622</td>
<td>451</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>16.77%</td>
<td>17.04%</td>
<td>14.50%</td>
<td>14.45%</td>
</tr>
<tr>
<td>XHTML</td>
<td>99,287</td>
<td>16,647</td>
<td>16,925</td>
<td>14,977</td>
<td>14,350</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>6,04%</td>
<td>6,64%</td>
<td>5,45%</td>
<td>5,06%</td>
</tr>
<tr>
<td>Humanoid</td>
<td>3,536</td>
<td>567</td>
<td>867</td>
<td>549</td>
<td>306</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>16.04%</td>
<td>24.52%</td>
<td>15.53%</td>
<td>8.65%</td>
</tr>
<tr>
<td>3D Map</td>
<td>35,219</td>
<td>2,078</td>
<td>2,444</td>
<td>1,665</td>
<td>1,648</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>5.90%</td>
<td>6.94%</td>
<td>4.73%</td>
<td>4.68%</td>
</tr>
<tr>
<td>Notebook</td>
<td>321</td>
<td>196</td>
<td>484</td>
<td>135</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>100.00%</td>
<td>61.06%</td>
<td>150.78%</td>
<td>42.06%</td>
<td>21.18%</td>
</tr>
</tbody>
</table>

Table 1: Compression Results Comparison

Of the two common desktop zip techniques, GZip delivered the best compression results. However, when compared to EXI (schemaless), in all cases other than X3D, EXI was approximately 10% less. When compared to EXI with schema, in all cases EXI was less, and nearly half the size of GZip.

Interestingly, the DIS XML format can be represented in EXI in slightly smaller size than the original binary format, due primarily to the use of variable-length numeric values. These has implications for the design of future DoD protocols; custom binary formats can be replaced by EXI encoded XML documents.

3.4 Summary of Schemaless Results

A comparison of the percentage of the original file size between GZip and EXI is shown in figure 9. It can be seen that EXI (other than X3D) is always smaller than GZip for the cases examined.
Since GZip is the most common compression technique employed in DoD M&S, it is the baseline comparison. Figure 10 shows the percentage EXI (schemaless) is of the same GZipped file. On average, the EXI (schemaless) file was 78% of the size of the GZIP.

Repeating the comparison with GZip as baseline, but this time with EXI schema informed, figure 12 shows that in every case the schema informed EXI compression resulted in a file size less than GZIP. On average the EXI compressed file was 63% of the size of the GZIP file.

### 3.6 W3C Corpus of Results

Additional results can be obtained from the EXI Working Group’s *Efficient XML Interchange Evaluation* document [7] and *Efficient XML Interchange Measurements Note* [6]. The results shown here agree with the findings of the EXI Working Group, which found that EXI was consistently (and often markedly) more compact than XML documents encoded with GZip.

### 4. Implementation issues

#### 4.1 Importance of a Schema

A schema enables the most compact representation of XML in the EXI format. Due to encoding limitations during the grammar learning process for a schemaless XML document, the level of compactness of schemaless EXI cannot surpass that of a schema informed document. However, the ability to encode XML without a schema is a W3C requirement because many XML documents do not have a schema, or if they do, existing XML documents do not fully comply with it. While schema-informed encoding does deliver better results, schemaless encoding generally delivers GZip or better level of compactness.

**N-bit minimization** The compact identifier encoding length is determined by the number of entries in a grammar, or by the size of a string table. Smaller values of compact identifiers are preferable because they can be represented in fewer bits.

Without a schema, the number of entries in a grammar is not known until EXI process completes because the
encoding algorithm is always learning until an EndDocument event is fired. The EXI encoding algorithm can exploit schema information to create grammars with smaller compact identifiers. A schema in essence creates the first occurrence of every element, attribute, and data type that will be encountered in an XML document before being processed by the EXI processor. This delivers optimized encoding immediately without the need for learning.

**Data Type Binding** A schema defines the data types of the XML document. EXI uses this information to write a binary format representation of numeric values that is usually smaller than the text representation.

Without a schema EXI encoding is limited to strings only; numeric, date, and other data types can only be represented as strings, though repeated values can be represented as a compact identifier. Retaining numeric values in a more compact binary format enables file size savings and reduced processor complexity. In addition, EXI often uses variable length binary representations for numeric data, so that, for example, integers with a small value can be represented in a single byte, and larger integer values are represented in multiple bytes.

### 4.2 Available EXI Implementations

There are a number of ongoing implementations developing in parallel at different levels of completeness.

The initial format authors, Agile Delta, have a commercial implementation in both Java and C++ [15].

Siemens Corporation has developed a free, open source EXI implementation in Java. It is licensed as GPL so the source code must be supplied, and any derivative works must also supply the source code under the same terms. Their implementation can be downloaded from sourceforge.net. [16].

The Naval Postgraduate School (NPS) is working on another implementation using the Apache open source license [openexi.sourceforge.net]. The Apache license is, in contrast to the GPL license, non-viral and friendlier to commercial applications. The ultimate goal is to release it as an Apache project.

### 4.3 Comparison Tool

The Naval Postgraduate School in its effort to evaluate the effectiveness of the EXI solution for DoD applications created a tool to compare EXI compactness to other common DoD compression formats.

**Figure 13: Technique Comparison Tool**

### 4.4 Options Tool

In addition to the compression comparison tool created at the NPS, a graphic user interface that exercises the EXI encoding options was built using the Siemens codebase engine.

**Figure 14: EXI Options Tool**

These tools, as well as NPS’s implementation progress can be downloaded freely from the NPS website located at: http://www.movesinstitute.org/exi/EXI.html

### 5. Conclusion & Recommendations

#### 5.1 Conclusion

The porting of data to XML is how IT is done across the gamut of IT domains. DoD is evolving towards XML as its primary data interchange standard, and in DoD M&S, XML is the primary data interchange format. Corporate America, and corporations around the world continue to utilize XML as the backbone of application data development, and the adoption grows constantly.

Under current computer architectures, XML has reached a distribution apex. The next generation of
computing devices such as mobile and micro devices cannot support native 1.x XML without a more compact and efficient representation due to their limited CPU, battery and memory capacity. Further, low bandwidth environments such as handheld devices or deployed DoD units, are unable to send and receive XML efficiently due to file size bloat. To enable the continued growth of XML into these environments, and the next generation of devices, an efficient and compact XML format is needed.

EXI is a general purpose format that has shown to work on the entire range of the XML family of languages. EXI is also on track to become the W3C recommended solution for standardization as a compact and efficient XML infoset encoding [5][6][11].

A standards based compact XML infoset format will enable any software vendor to support any client’s XML needs without proprietary format considerations. Standardized format ensures XML’s continued interoperability, the cornerstone of XML’s incredible worldwide success.

A domain case could build a slightly more compact technique for its domain specific uniqueness. X3D for example knows that every file will have the tag set <SCENE>, so an X3D specific compression could disregard those tags at compression because the decompression routine could add it by implicitly. However, any domain case that creates its own domain specific compression techniques will not support the entire family, and will not be interoperable with the entire XML stack.

EXI delivers roughly a doubling of bandwidth utilization in compactness when compared to the most common (desktop, web server, other) compression technique.

5.2 Recommendations

EXI should be built into the web server architecture:
1. XHTML is the future of web pages and is an XML language
2. HTTP port 80 is the only reliable port open from any source to destination, so most future file transfers will be conducted through a web server.

Tools that process XML should consider EXI as their disk file format. Few truly edit XML outside of an IDE, so let the IDE do the binary to string human readable conversions, while EXI performs the compact storage.

EXI development should include an open source implementation with an open, non-viral license. Many applications would benefit form an embedded EXI library, and commercial or open source viral-licensed code are often problematic to incorporate in other projects.

The purpose of EXI is to enable the entire family of XML languages to grow deeper into the web.

6. References

Author Biographies

DON BRUTZMAN is a Modeling, Virtual Environments and Simulation (MOVES) Associate Professor at the Naval Postgraduate School in Monterey, CA.

DON MCGREGOR is a Modeling, Virtual Environments and Simulation (MOVES) Software Engineer at the Naval Postgraduate School in Monterey, CA.

SHELDON SNYDER is a Modeling, Virtual Environments and Simulation (MOVES) student at the Naval Postgraduate School in Monterey, CA.