Schema of the Machine Readable Enumerations Document

William Oliver
Peter W. Ross
Defence Science and Technology Organisation
506 Lorimer Street
Fishermans Bend, VIC 3207
Australia
+61 3 9626 7000
{william.oliver,peter.ross}@dsto.defence.gov.au

Keywords:
Distributed simulation, enumerations, schema

Abstract: *An essential resource for those responsible for developing and configuring distributed simulations is SISO-REF-010 Enumerated and Bit-encoded Values. This Microsoft Word document defines numeric values, known as enumerations, for a large catalogue of military and commercial equipment. As part of a broader effort to improve the management of SISO-REF-010, a schema has been developed to represent enumerations in the Extensible Mark-up Language (XML). This paper describes the first release of the machine enumeration document schema, and provides some historical context.*

1 Introduction

In distributed simulations, data is exchanged before, during and after the simulation. Distributed Simulation protocols, such as DIS and HLA, address the information exchange during the execution of the simulation but have little guidance on information exchange before or after a simulation. Usually any information exchange in these phases is performed manually.

One such manual data exchange is initialising simulation enumerations. As simulation exercises become more frequent and increase in size, this initialisation process becomes a constraint on productivity, and a barrier to more responsive and even larger simulations.

A machine readable version of SISO-REF-010[1], has been developed to help alleviate this issue and create a more usable and maintainable document for the future.

This machine readable document permits a single source of data for all simulators, as well as a human readable version. A machine readable format also allows a degree of code simplification, as a single API for accessing enumerations data can be used on many simulators.

This paper describes the schema of the machine readable enumerations document.

2 History

The original enumerations document existed in Microsoft Word format from 1992 to 2010. This format is not easily amenable to automatic processing. Additionally it consists largely of free text and the formatting is inconsistent through the document, making it difficult to process.

There have been at least five calls for a machine readable enumerations document [1], [2], [3], [4], [5]. Four benefits of having a machine readable document have been noted [6]:

1. Allows the document to be loaded directly by software, removing transcription errors.
2. Permits meta-data to be freely associated with the data.
3. Improves revision control. MS Word is a binary format and differences between releases are difficult to see.

[1] The previous title 'Enumerations and Bit-Encoded Values' — often just 'EBV-DOC' has been changed to 'Enumerations for Simulation Interoperability'

4. Permits efficient viewing (or processing) of part or all of the document as required. For example, layout of the document may be changed.

2.1 Machine Readable Formats

There have been at least two published attempts at creating a machine readable format.

The Institute for Simulation and Training produced the 'DIS Data Dictionary', a database of entity-type enumerations and DIS message structures. It was a Microsoft Access database and exploited the report and query tools provided by Microsoft Access and compatible database engines. An HTML version was available on the Internet [7], though the database content has not been updated since 1996.

In 2005 the following requirements for a machine readable version of the enumerations document were proposed [8]:

1. Both a human readable document and at least one machine readable format are required;
2. there must be a process to convert between the human readable document and the machine readable format;
3. any conversion tools that support this process must be available at no-cost (or industry standard), and not be encumbered by restrictive licensing conditions;
4. any tools must not have a steep learning curve or require expert level knowledge to use; and
5. any tools must be available for Microsoft Windows.

The enumerations maintainer then released an XML document named 'XML encoding of DIS EBV' (XoDIS), together with a partially populated XML data file containing entity-type and value-pair enumerations.

2.2 An Improved Format

A third machine readable format was published in 2007. This improved on the XoDIS proposal, as it offered a fully populated XML data file, encouraging more immediate adoption. It also used a simpler, more general, information schema that did not have to be updated each time tables were added or removed from the document. Both the XoDIS requirements and the 2007 proposal have formed the basis of the SISO-REF-010 XML document.

2.2.1 Which Technology to Use (or Why XML)?

The main purpose is to have a single source of information that can be made easily accessible by both humans and computers. There are two approaches that can be taken, have a *human* readable master format that can be processed by a computer, or a *computer* readable master that can be processed to be human readable. The XoDIS requirements noted previously, called on MS Windows availability, however many simulations run on older workstations, some on unusual machines. Recent simulations typically run on x86 or amd64 compatible computers, and thus it is also essential that the technology be cross platform. Technologies that have been proposed include Structured Query Language (SQL), Microsoft Access, the Lightweight Directory Access Protocol (LDAP) and plain Comma Separated Value files (CSV).

The purpose of the enumerations document is to act as a repository for simulation enumerations data: it should not put any constraints on how it is used and implemented. This means that it *must* be convertible into a format acceptable to the end user. This implies the requirement that the machine readable document be easily converted to other formats.

The decision to use XML came from the fact that of all the technologies suggested,

- XML is an open standard from the World Wide Web Consortium (W3C);
- XML is widely used and cross platform;
- Its openness and ubiquity ensure there are many tools, both free and expensive, open and proprietary;
- Unlike LDAP or SQL it requires no server software;
- It has a form for describing transformations to other forms (XSLT—which is itself an open standard);
- XML is a development of older and proven technologies (both SGML and HTML);
- It is a text format,
  - It is readable by both humans and computers (although raw XML is not ideal for humans);
  - There are many revision control systems and difference tools to compare versions.

2.2.2 Which information model to use (or why not XoDIS)?

The original enumerations document specified a total of 279 tables, where each table lists enumerated values for the various DIS Protocol Data Unit (PDU) fields. These tables were grouped together and arranged into sections. Each section begins with a brief description of the table, followed by the table itself.

The XoDIS draft incorporated the notions of section hierarchy into its information schema, and proposed unique XML elements for each enumeration table. This created an XML document with over 279 different elements. Whilst this captured the content and structure of the enumerations document it was felt that the grouping of data and presentation into one was not an optimal design choice, and the number of elements would make programming difficult.

There are four distinctive table categories used in the document, namely:

1. standard value-description pairs (see table 1), and
2. bit-masks describing the layout of bit-fields (see table 2).
3. entity-type enumerations (see table 3),
4. object-type enumerations (similar to entity-type enumerations, but not as deeply nested).

Field Value	Function Description
1	Multi-function
2	Early Warning
3	Height Finding

Table 1: Example of a Name-Value pair.

Name	Bits	Purpose
Damage	3–4	Damaged appearance of an air entity
		0 - No damage
		1 - Slight damage
		2 - Moderate damage
		3 - Destroyed

Table 2: Example of a Bit-Mask enumeration.

The document also specifies data record structures, which are specific to the DIS protocol. It is recognised [9], that data record structures do not belong in the enumerations document and a separate document has been proposed by the enumerations coordinator to store such records.

The data model chosen for the machine readable enumerations document was to have distinct elements for the four categories of data shown above. The perceived benefits are that a small schema is more easily memorised so a programmer can use it without having to refer constantly to documentation, and that the amount of software written to process enumerations data is reduced — that is, the same structures and algorithms can be used on a greater subset of all enumerations data.

3 General Concepts

One of the main considerations at the design stage was to keep the schema as compact as possible. The intent was to make all like elements of the same type. It is less effort to write software to handle a generic enumeration than to have to write one to handle every case.

There are two information models one must consider in the XML schema. One is the information model of the data (the contents of the enumerations document), the other is the information model of the schema itself (which is expanded on in section 4).

3.1 XML Elements

As we are replicating the existing enumerations document, the data model mirrors the way the content is presently arranged. The information falls into a few general categories, in addition to the types of data listed in section 2.2.2 there is document meta-data. It represents information about the document itself and it is generally not intended for end users of simulations.

Standard value-description pairs (often called enumerations), are described by the `enum` element, which contains `enumrow` elements, which define individual *rows* in the table. This approach was designed to allow the addition (or removal) of tables of enumerations without requiring the schema to change as well. The example shown in table 1 would be represented by the following simplified XML block:

```
<enum>
  <enumrow value="1"
     description="Multi-Function"/>
  <enumrow value="2"
     description="Early Warning"/>
  <enumrow value="3"
     description="Height Finding"/>
</enum>
```

An entity type table consists of all entity types that share a kind, domain, and country (for example, there is a table for all Australian surface platforms). The example entity type table shown in table 3 would be represented by the following XML block:

```
<cet>
  <entity kind="1" domain="3" country="13">
    <category value="6"
      description="Guided Missile Frigate">
      <subcategory value="1"
        description="ANZAC Class (Meko 200)">
        <specific value="1"
          description=''FFH 150 ANZAC''>
      </subcategory>
    </category
```

Kind	Domain	Country	Category	Subcategory	Specific
1	3	13			
			6 Guided Missile Frigate		
				1 ANZAC Class (Meko 200)	
					1 FFH 150 ANZAC

Table 3: Example of an entity type enumeration.

```
  </entity>
</cet>
```

Bitmasks can contain a number of enumerated values. The example of a bitmask enumeration in table 2 would be represented as the following XML block:

```
<bitmask>
  <bitmaskrow_range
    name="Damaged appearance of an air entity"
    value_min="3"
    value_max="4">
    <enumrow value="0"
      description="No Damage"/>
    <enumrow value="1"
      description="Slight Damage"/>
    <enumrow value="2"
      description="Moderate Damage"/>
    <enumrow value="3"
      description="Destroyed"/>
  </bitmaskrow>
</bitmask>
```

Note how the `bitmaskrow_range` element contains `enumrow` elements. This is because they share the same properties as an individual datum in a regular enumeration, reusing the element here reduces the number of elements we require.

Figure 1 and the simplified structure below represent the element hierarchy. They show the XML elements that make up the machine readable enumerations document and their hierarchical relationship to each other. The presence of a `(_range)` in the XML shown below denotes that there is an alternative element (with `_range` appended—for example there is both an `enumrow` and a `enumrow_range` element) that allows for a range of values).

```
ebv
 +- revisions
 |    +- revision
 +- dict
 |    +- dictrow
 +- enum
 |   +- header
 |       +- col
 |   +- enumrow(_range)
 |       +--meta
 +- bitmask
 |   +- bitmaskrow(_range)
 |       +- enumrow(_range)
 |           +- meta
```

```
+- cet
|   +- entity
|       +- category(_range)
|          +- subcategory(_range)
|             +- specific(_range)
|                +- extra(_range)
+- cot
|   +- object
|       +- category(_range)
|          +- subcategory(_range)
+- record
    +- field
        +- datatype
```

3.2 Unique Identifiers

Every value in DIS can be uniquely identified by its location in a PDU, and its value can be looked up in the enumerations document. This process can be cumbersome and specific to DIS. In order that there be simpler methods for identifying an individual datum, and the enumerations made more easily accessible by other protocols (such as HLA through the RPR-FOM) a number of unique identifiers are used in the schema.

Every table in the machine readable enumerations document has a *Unique Identifier* (UID—which is an integer). This allows the referencing of any table in software without having to perform string comparisons. The assigned *uid* for a table will not change [10].

Each row within a table is assigned a *Universally Unique Identifier* or UUID [11]. These are 128-bit numbers guaranteed to be unique both within the document and *universally* unique.

Entity type rows are also assigned a *uid*, starting at $10,000$. These values are provided explicitly to support compatibility with the Common Image Generator Interface (CIGI), which requires a 32-bit identifier.

4 XML Schema Design Detail

At its most simple, XML schemas define the elements and attributes that can appear in an XML document, and place constraints upon the contents. XML has a type system to

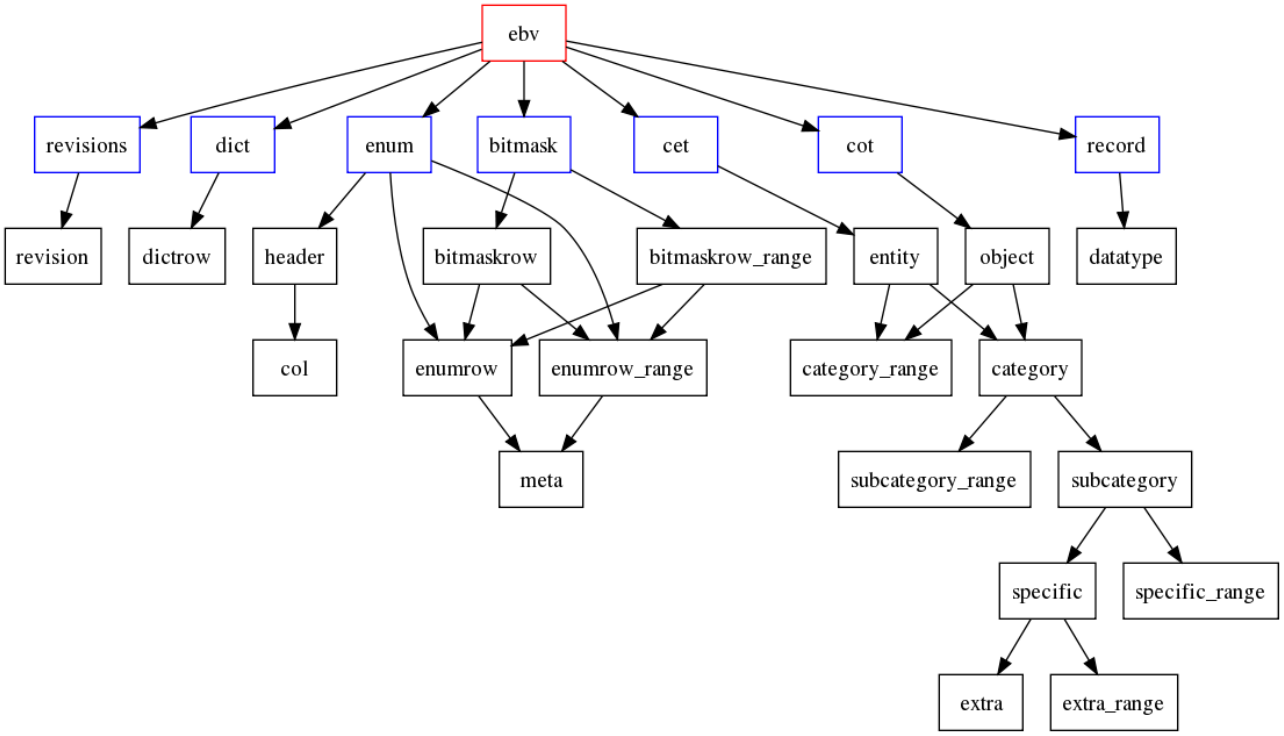

Figure 1: The machine readable enumerations document XML hierarchy. Note that `cr` and `cr_range` elements are not shown as they can be children of any element (except the root element).

allow the reuse and extension of elements, simplifying the design of the schema.

The XML schema hierarchy has a number of generic abstract types, which define most of the attributes. Elements that appear in the machine readable enumerations document are *instances* of types that inherit from one of these generic types.

The information model and XML elements introduced in section 3 are expanded on below.

4.1 Schema Generic Types

The schema defines a number of *abstract* types. An abstract type in XML cannot be used as an instance in a document. The purpose is to define attributes or restrictions that are common to a number of elements, such that another XML type can then be *derived* from the abstract type and inherit its properties as well as add its own. The abstract types of the machine readable enumerations document are:

genericentry_t This is the base type for all elements that define a table. Its attributes are shown in table 5.

genericentry_t This is the base type for all entities and objects. Its attributes are shown in table 4.

genericentrydescription_t extends `genericentry_t` by adding a mandatory attribute `description` (a string).

genericentrystring_t is the base type for all single valued entries that are strings. It extends `genericentrydescription_t`, adding a mandatory attribute `value` (a string).

genericentrysingle_t is the base type for all single valued entries that are integers. It extends `genericentrydescription_t`, adding a mandatory attribute `value` (an integer).

genericentryrange_t is the base type for all range values that are integers. It extends `genericentrydescription_t`, adding the attributes `value_min` and `value_max` (integers).

4.2 Document Root

All XML documents are required to have a single root element [12]. The root element for the machine readable enumerations document is `ebv`. Its attributes are shown in table 6 and the hierarchy has been shown in section 3 and graphically in figure 1.

Name	Type	Use	annotation
footnote	string	optional	Any additional information pertaining to the enumeration entry.
xref	positiveInteger	optional	Cross-reference to another enumeration table (uid).
deprecated	boolean	optional	Flag to indicate the enumeration entry has been deprecated.
status	string	optional	Flag to indicate the approval status of the entry. *Pending* denotes that the entry has been proposed, but not yet approved by the EWG; *New* means that the entry has been approved by the EWG since that last formal issue of the database.
draft1278	boolean	optional	Flag to indicate the enumeration entry applies to a draft revision of IEEE 1278
uuid	uuid	required	Unique numeric identifier for the enumeration entry (RFC-4122).
baseuuid	boolean	optional	Indicate an enumeration entry UUID that this entry was based on (RFC-4122).

Table 4: Attributes of the abstract type `genericentry_t`

Name	Type	Use	annotation
uid	positiveInteger	required	Unique numeric identifier for the enumeration table.
name	string	required	Name of the table.
draft1278	boolean	optional	Flag to indicate the enumeration table applies to a draft revision of IEEE 1278.
deprecated	boolean	optional	Flag to indicate the enumeration table has been deprecated.

Table 5: Attributes of the abstract type `generictable_t`

4.3 Document Meta-Data

The meta-data elements record information about the enumerations document. There are three kinds of meta-data:

1. **revision** elements that provide a mechanism to record information about previous published versions of the document. There are two elements of note, `revisions` and `revision`; the former is the container element for the list of revisions, the latter represents an individual revision.

2. **change request** elements support revision control. All elements within the schema, except the root element, may contain the `cr` or `cr_range` elements. These elements permit changes to be linked back to the originating SISO-REF-010 change request form.

3. **dictionary** elements are used to maintain a list of acronyms. The `dict` element defines the table and **dictrow** elements represent an acronym and its definition.

The meta-data elements of the machine readable enumerations document are:

`revisions` extends the `generictable_t` by allowing `revision` elements as children.

`revision` has three attributes:

`title` Title of the document—a mandatory attribute (a string).

`date` Publication date specified in ISO 8601 date format (YYYY-MM-DD)—an optional attribute.

`uuid` Unique numeric identifier for the revision entry—a mandatory attribute.

`cr` has one mandatory attribute `value`, the change request number (an integer).

`cr_range` has two mandatory attributes `value_min` and `value_max`, the minimum and maximum change request numbers (integers).

`dict` extends the `generictable_t` by allowing `dictrow` elements as children.

`dictrow` is an instance of `genericentrystring_t`. It contains an acronym and its definition.

4.4 Enumerations Table

The `enum` element is used to represent a table of standard name-description pairs. These are the most frequent table types in the standard.

`enum` is an instance of `enum_t` which extends `generictable_t` by allowing `header`, `enumrow` and `enumrow_range` elements as children.

`header` allows column headings for the table. It has `col` elements as children.

`col` has two attributes:

Name	annotation
title	Title of the document (e.g. SISO-REF-010-2010.1).
release	Release or version number (e.g. Draft 3, Final).
href	Internet hyperlink where this document is published.
date	Publication date specified in ISO 8601 date format (YYYY-MM-DD).
description	Description of the document.
organisation	Publishing organisation.

Table 6: Attributes of the Root Element (<ebv>).

key Meta-data name associated with the column—a mandatory attribute (a string).

name Name of the column—a mandatory attribute (a string).

enumrow extends the type genericentrysingle_t by allowing meta elements as children.

enumrow_range extends the type genericentryrange_t by allowing meta elements as children.

meta permits arbitrary meta-data to be associated with an enumeration. It has two attributes

key Meta-data name or identifier (e.g. units)—a mandatory attribute (a string).

value Meta-data value (e.g. meters per second)—a mandatory attribute (a string).

4.5 Bitfield Table

A bit-encoded value in DIS is a data structure where a number of enumerations are packed into a range of bits in a field; that is a bitmask contains a number of enumeration values. In the enumerations document a bitfield table shows the enumerations contained in a bit-encoded value.

Bitfield tables are represented by the element bitmask while individual bitfields are represented by bitmaskrow which contains enumrow (or enumrow_range) elements to represent an individual enumeration value (or row) in the table.

bitmask extends the type genericentrysingle_t by allowing child elements bitmaskrow and bitmaskrow_range.

bitmaskrow extends the type genericentrysingle_t by allowing enumrow and enumrow_range elements as children.

bitmaskrow_range extends the type genericentryrange_t by allowing enumrow and enumrow_range elements as children.

4.6 Comprehensive Entity Type

The cet element is used to represent the base of the comprehensive entity type tables. In the schema the cet element contains a list of entity elements. These correspond to the tables in the previous EBV-DOC, that is, they are uniquely defined by a combination of kind, domain and country. Beneath the entity element there is a hierarchical structure of category, subcategory, specific and extra elements that uniquely define an entity.

cet extends genericentrysingle_t by allowing entity elements as children.

entity is an instance of entity_t which extends genericentry_t by allowing category and category_range elements, and the following attributes:

kind a mandatory integer.

domain a mandatory integer.

country a mandatory integer.

uid Unique numeric identifier.

category extends the type genericentrysingle_t by adding the mandatory attribute uid (an integer) and allows the child elements subcategory and subcategory_range.

category_range extends the type genericentryrange_t by adding the mandatory attribute uid (an integer) and allows the child elements subcategory and subcategory_range.

subcategory extends the type genericentrysingle_t by adding the mandatory attribute uid (an integer) and allows the child elements specific and specific_range.

subcategory_range extends the type genericentryrange_t by adding the mandatory attribute uid (an integer) and allows the child elements specific and specific_range.

specific extends the type genericentrysingle_t by adding the mandatory attribute uid (an integer) and

allows the child elements `extra` `extra_range`.

specific_range extends the type `genericentryrange_t` by adding the mandatory attribute `uid` (an integer) and allows the child elements `extra` `extra_range`.

extra extends the type `genericentrysingle_t` by adding the mandatory attribute `uid` (an integer).

extra_range extends the type `genericentrysingle_t` by adding the mandatory attribute `uid` (an integer).

4.7 Comprehensive Object Type

Comprehensive object types are described in the `cot` element. The layout and attributes of these types are identical to comprehensive entity types.

cot extends `genericentry_t` by allowing `object` elements as children.

object is an instance of `object_t` which extends `genericentry_t` by allowing `category` and `category_range` elements, and the following attributes:

kind a mandatory integer.

domain a mandatory integer.

uid Unique numeric identifier.

5 Transformation Tools

Closely related to the schema are tools to transform the document into human-readable and application specific formats. XML has its own transformation language, XSL (eXtensible Stylesheet Language [13]). A number of transformations have been developed.

1. HTML transform

   The HTML transformation allows the document to be rendered by a web browser and appears similar in structure to the existing MS Word version of the enumerations document. It includes Microsoft Word (WordML) specific mark-up. This allows the HTML document to be loaded into Microsoft Word, with table of contents, page headers and footer displayed correctly. At the time of writing this transformation is no longer included with SISO-REF-010.

2. SpreadsheetML transform

   The SpreadsheetML transform uses Microsoft's XML schema for spreadsheets to allow the creation of a Microsoft Excel compatible spreadsheet. The spreadsheet consists of a worksheet for every table in the enumerations document as well as an index page to allow easier navigation.

3. C99 header transform

   This transform outputs a list of table identifiers, allowing programs written in the C, C++ or Objective-C languages to request a table from the XML document.

These transforms are provided as proof of concept that the XML schema allows relatively easy re-use of the data in a variety of convenient forms. Others may be written, for example, to generate a format specific to a simulator.

6 Criticisms

6.1 Choice of Schema Languages

There is no requirement that an XML document have a schema. XML has a concept of 'well-formed' that is, it conforms to the XML syntax rules but places no restrictions on the elements and attributes allowable in the document. There are two main schema languages for XML: RELAX-NG and the W3C XML Schema Language. The W3C schema language was chosen to stay with the W3C's recommendations. Further reading suggests that the W3C schemas are better supported at this time, although less powerful than RELAX-NG [14].

6.2 Elements vs Attributes

There do not appear to be any hard and fast rules about the merit of attributes versus elements in XML documents. One main difference is that an attribute may appear only once in an element. It is therefore more cumbersome to have multiple values for an attribute but trivial to have multiple values for an element. With regard to the enumerations document most information is encoded in attributes. Having looked extensively at other XML languages the reverse seems to be more prevalent.

6.3 The Lack of Text Nodes

In XML terminology text nodes are what goes between an opening and closing tag of an element.

```
<element attribute="value">
    This is the text node
```

```
</element>
```

The XML enumerations document has no data as text nodes; all data is encoded as attributes of elements, rather than as text nodes enclosed by elements. This was not a conscious design decision but came about as the authors of the schema simply chose to encode the information in this way.

6.4 The Lack of Descriptive Text

The XML enumerations document does not contain the descriptive text before and/or after tables that appear in the MS Word version. The descriptive text exists at present in the templates to which the various transform tools write their data. This is a potential problem as it means there are two sources of data. It is generally good practice to separate data and presentation. In fact, the XML languages chosen make this explicit, XML for the data and XSL for the presentation logic. Having the enumerations and the text describing enumeration tables, in separate documents (the XML and the XSL documents) means there are potentially two places to edit data.

When initially embarking on developing the machine readable enumerations document, a goal was to create a human readable version that mimicked the existing one as closely as possible. At the time the only way to achieve this was to put the text in the XSL script.

In future it should be possible to put this text into the XML file and remove it from the XSL file, providing a small number of style changes can be made to the human readable form.

6.5 The use of Typename_t to Describe Types

The schema makes extensive use of abstract types to define the elements. For example, the *enum* element is an instance of the *enum_t* type. The use of *_t* at the end of a type name is a convention the authors adopted from the *C* programming language.

While this is perfectly valid, it is not the usual XML convention. It is more common to see some variation of the so called *camelcase*, where words are strung together and the first letter is capitalised. So, our *enum* element would be an instance of *enumType* or *EnumType*. As with all style conventions, consistency is usually more important than the convention chosen.

7 Conclusion

In this paper we have detailed the development history of the machine readable enumerations document and described the schema in terms of its constituent elements and attributes. Having started with the requirements set forth in XoDIS it is hoped this evolution of the document has produced a schema that is both relatively easily understood, yet comprehensive enough to capture all the enumerations data and see widespread adoption in the simulation community. Its first formal use is to be the next release of SISO-REF-010.

8 References

[1] G. Sauerborn, "Non word version?," 1997. http://discussions.sisostds.org/default.asp?action=9&read=14433&fid=162&boardid=2.

[2] T. DeCarlo, "Looking for enum doc in MS Access format," 1999. http://discussions.sisostds.org/default.asp?action=9&read=14488&fid=163&boardid=2.

[3] B. Clay, "DIS enumerations list," 2003. http://discussions.sisostds.org/default.asp?action=9&read=14915&fid=167&BoardID=2.

[4] G. Shanks, "Machine readable DIS Enumerations," 2005. http://discussions.sisostds.org/default.asp?action=9&read=15143&fid=31&BoardID=2.

[5] D. Bruztman, "DIS Enumeration Database and XML," 2005. http://discussions.sisostds.org/default.asp?action=9&read=15188&fid=31&BoardID=2.

[6] P. Ross, "Machine readable enumerations for improved distributed simulation initialisation interoperability," in *SimTecT 2008 Conference Proceedings*, (Melbourne, Australia), Simulation Industry Association of Australia, Simulation Industry Association of Australia, 12–15 May 2008.

[7] Institute for Simulation and Training, "DIS Data Dictionary (on-line version)." http://siso.sc.ist.ucf.edu/dis-dd.

[8] G. Shanks, "Machine Readable Enumerations - DIS Product Development Group Meeting," in *Fall Simulation Interoperability Workshop*, no. 05F-SIW-212, (Orlando, Florida, USA), September 2005.

[9] L. Marrou, "Enumerations issues," 2009. http://discussions.sisostds.org/threadview.aspx?fid=32&threadid=46906.

[10] Distributed Interactive Simulation Product Support Group, *SISO-REF-010.1-2010 Operations Manual for the Distributed Interactive Simulation Product Support Group Enumerations Working Group*. Simulation Interoperability Standards Organization, December 2010.

[11] P. Leach, M. Mealling, and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace." RFC 4122 (Proposed Standard), July 2005.

[12] E. Maler, J. Cowan, J. Paoli, F. Yergeau, T. Bray, and C. M. Sperberg-McQueen, "Extensible markup language (XML) 1.1 (second edition)," W3C recommendation, W3C, Aug. 2006. http://www.w3.org/TR/2006/REC-xml11-20060816.

[13] M. Kay, "XSL transformations (XSLT) version 2.0 (second edition)," W3C proposed edited recommendation, W3C, Apr. 2009. http://www.w3.org/TR/2009/PER-xslt20-20090421/.

[14] T. Bray, "Choose Relax Now," 2006. http://www.tbray.org/ongoing/When/200x/2006/11/27/Choose-Relax.