Applying a subset of the UBL NDR to the UnitsML schema

Ronny Jopp [1, 2]

Alexander Roth [1, 2]

[1]Biochemical Science Division, National Institute of Standards and Technology, Gaithersburg, MD, U.S.A

[2]Computer Science Department, University of Applied Sciences, Wiesbaden, Germany

1. Introduction

This document delineates the naming and design rules (NDRs) for the Units Markup Language (UnitsML) schema [4]. In keeping with the spirit of component reuse in the eXtensible Markup Language (XML) and further the creation of an OASIS technical committee UnitsML goes along with a subset of the NDR defined for the OASIS Universal Business Language (UBL) [4].

2. Recommended Rules for UnitsML

2.1. Relations to standards

The W3C XML Schema Definition Language (XSD) has become the generally accepted and most widely adopted schema language. Accordingly, the NDR for UnitsML and the UnitsML schema must conform to the W3C XSD requirements.

> [STA1] The UnitsML schema design rules MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.

> [STA2] The UnitsML schema MUST be based on the W3C suite of technical specifications holding recommendation status.

2.2. Overall schema structure

These schema structure issues must be addressed to provide a concise look and feel to the UnitsML schema that will facilitate the integration of UnitsML as a standard among the XML community and use on other XSD-based markup languages.

[GXS1-UnitsML] The UnitsML Schema MUST conform to the following physical layout as applicable:

```
<!-- ======= XML Declaration======= -->
<?xml version="1.0" encoding="UTF-8"?>
<!-- ===== Copyright Notice ===== -->
      Required OASIS full copyright notice.
```

```
<!-- ===== xsd:schema Element With Namespaces Declarations ===== -->
xsd:schema element to include version attribute and namespace
declarations in the following order:
      xmlns:xsd
      Target namespace
      Default namespace
      Unqualified Datatypes
      Qualified Datatypes
      Identifier Schemes
      Code Lists
      Attribute Declarations – elementFormDefault="qualified"
      attributeFormDefault="unqualified"
      Version Attribute
<!-- ===== Imports ===== -->
      all incorporated schemas
<!-- ===== Root Element ===== -->
      Root Element Declaration
      Root Element Type Definition
<!-- ===== Element Declarations ===== -->
      ~~alphabetized order~~all element declarations
<!-- ===== Element Type Definitions ===== -->
      ~~alphabetized order~~all complex type definitions
<!-- ===== Attribute Type Definitions ===== -->
      ~~alphabetized order~~all global attribute (group) definitions
<!-- ===== simpleType Type Definitions~~Copyright Notice~~ ===== -->
      ~~Required OASIS full copyright notice.~~all simple type definitions
```

Mixed content is intended for use in document-centric schemas. UnitsML is a data-centric format, and stored pieces of data must be clearly unambiguous. The use of white-space in mixed content makes processing unnecessarily difficult.

> [MDC2] Mixed content MUST NOT be used except where contained in an
> `xsd:documentation` element.

2.3. Reusability

To maximize reusability of elements declared within the UnitsML schemas, all element declarations must be global. This type of element declaration enables the re-use of elements both inside and outside of the UnitsML standard. Global declarations enable design flaws in schema implementations to be discovered more easily and avoided in future revisions.

> [ELD2] All element declarations MUST be global

Annotation:

Since every global declared element can act as a root element, it is not possible to enforce the use of the correct root with standard tools. I think this is problematic and we should discuss that issue. A possible solution could be to create a Schematron file to ensure the use of the proper root element `UnitsML` for standalone UnitsML documents.

Annotation#2:

This is not really a problem though, as only one of the intended uses for UnitsML is to provide instance document document roots. Usage of fragments of UnitsML only is a valid use-case for UnitsML, too.

2.4. Namespaces

XML namespaces provide a simple method for qualifying element, types and attribute names used in XML documents by associating them with namespaces identified by URI (uniform resource identifier) references [4]. Namespaces are developed for modularity and to enable the re-use of well-understood markup vocabularies. Declared elements,

attributes or types in one schema can be qualified by assigning a namespace prefix that binds a namespace to this prefix.

[NMS1] The UnitsML schema MUST have a namespace declared using the `xsd:targetNamespace` attribute.

Schema validation ensures that an instance conforms to its declared schema. There must never be two (different) schemas with the same namespace URI. In keeping with rule [NMS1], each UnitsML schema shall be part of its own versioned namespace.

[NMS2] Every UnitsML schema version MUST have its own unique namespace.

For interoperability and reusability of UnitsML in other schemas, and the use of external schemas within UnitsML, a consistent approach to namespace declarations is necessary.

[NMS3] The UnitsML namespaces MUST only contain the UnitsML schema.

To ensure conformance with the organizational structure of OASIS this structure must be mapped within the namespace name. The syntax for OASIS namespace names is described in IETF's RFC3121 [4]. This implies the use of a persistent URN instead of a resolvable URL.

[NMS4] The namespace names for UnitsML Schemas holding committee draft status MUST be of the form:
`urn:oasis:names:tc:unitsml:schema:<subtype>:<document-id>`

Example:
`urn:oasis:names:tc:unitsml:schema:xsd:UnitsMLSchema-1.0`

[F] The namespace names for UnitsML Schemas holding OASIS Standard status MUST be of the form:
`urn:oasis:names:specification:unitsml:schema:<subtype>:<document-id>`

Example:
`urn:oasis:names:specification:unitsml:schema:xsd:UnitsMLSchema-1.0`

Once a UnitsML namespace name is officially published, it must never be changed to maintain the persistence inherent with using URNs to define UnitsML namespace names.

[NMS6] A published UnitsML namespace MUST never be changed.

If two namespaces are mutually dependent then clearly, importing one will cause the other to be imported as well. For this reason, circular dependencies must not exist between external schemas imported into the UnitsML schema. By extension, circular dependencies must not exist between namespaces. A namespace "A" dependent upon type definitions or element declarations defined in another namespace "B" must import "B's" document schema.

[SSM2] A schema in a UnitsML namespace that is dependent upon type definitions or element declarations defined in another namespace MUST only import the document schema from that namespace.

2.5. Versioning

Versioning enables tracking of changes in schema files. It provides for better maintainability as older revisions can be distinguished from the current version.
The versioning is encoded in two different ways – firstly in the namespace name, secondly in an attribute `version` for the XSD root element `xsd:schema`. Each must represent the same version of a particular schema. To ensure compatibility and improve interoperability on the schema level, a versioning concept encoded in the namespace names is necessary.

[VER1] Every UnitsML schema major version committee draft MUST have an RFC 3121 conform `document-id` of the form: `<name>-<major>.0[.<revision>]`

Example:
`urn:oasis:names:tc:unitsml:schema:xsd:UnitsMLSchema-1.0.1`

[VER2] Every UnitsML schema major version OASIS Standard MUST have an

RFC3121 conform `document-id` of the form:

`<name>-<major>.0`

Example:

`urn:oasis:names:specification:unitsml:schema:xsd:UnitsMLSchema-1.0`

[VER3] Every minor version release of a UnitsML schema draft MUST have an

RFC3121 conform `document-id` of the form:

`<name>-<major >.<non-zero>[.<revision>]`

Example:

`urn:oasis:names:tc:unitsml:schema:xsd:UnitsMLSchema-1.1.1`

[VER4] Every minor version release of a UnitsML schema OASIS Standard MUST

have an RFC 3121 document-id of the form

`<name>-<major >.<non-zero>`

Example:

`urn:oasis:names:specification:unitsml:schema:xsd:UnitsMLSchema-1.1`

Once a schema version is assigned a namespace, that schema version and that namespace
will be associated in perpetuity. Any change to any schema module mandates association
with a new namespace.

[VER5] For UnitsML Minor version changes `<name>` MUST not change,

To differentiate the logical progressing of the schema evolution the major and minor
version numbers are positive integer incremented by one for major and minor changes.

[VER6] Every UnitsML Schema major version number MUST be a sequentially

assigned, incremental number greater than zero.

Example:

Old: `urn:oasis:names:specification:unitsml:schema:xsd:UnitsMLSchema-1.0`

New: `urn:oasis:names:specification:unitsml:schema:xsd:UnitsMLSchema-2.0`

> [VER7] Every UnitsML Schema minor version number MUST be a sequentially
> assigned, incremental non-negative integer.

Example:

Old: `urn:oasis:names:specification:unitsml:schema:xsd:UnitsMLSchema-1.1`

New: `urn:oasis:names:specification:unitsml:schema:xsd:UnitsMLSchema-1.2`

2.6. General Naming Rules

The official language for UnitsML is English. To avoid ambiguities, all official XML constructs in UnitsML must be in English using American spellings. The OED is the definitive dictionary of the English language. This multi-volume treatise is also available on CD ROM and on line (by subscription), as well as in simplified editions such as the Oxford English Dictionary for Writers and Editors.

> [GNR1] UnitsML element, attribute, and type names MUST consist of at least one
> word that MUST be in the English language, as found in the Oxford English Dictionary
> (OED) (Latest Ed.). Where both American and English spellings of the same word are
> provided, the American spelling MUST be used.

Annotation:

Rule GNR1 was altered with the restriction to the American spelling of the OED as found in the NDR created by the Department of the Navy (DON NDR Rule GNR1) [4].

Acronyms and abbreviations affect semantic interoperability and, as such, are to be avoided to the maximum extent practicable. Since some abbreviations will inevitably be necessary, UnitsML will maintain a normative list of authorized acronyms and abbreviations. Appendix A provides the current list of permissible acronyms, abbreviations, and word truncations. The intent of this restriction is to facilitate the use of

common semantics and to promote greater understanding. Appendix A will be updated to reflect growing requirements.

[GNR4] UnitsML XML element, attribute, and simple and complex type names MUST NOT use acronyms, abbreviations, or other word truncations, except those in the list of exceptions published in Appendix A.

The UnitsML standard does not desire a proliferation of acronyms and abbreviations. Appendix A is an exception list and will be tightly controlled by NIST's UnitsML working group (OASIS UnitsML TC in the future). Additions will only occur after careful scrutiny to include assurance that such addition is critically necessary, that acronyms and abbreviations have not already been declared in another UnitsML document, and that such an addition will not create semantic ambiguity of any kind.

[GNR5] Acronyms and abbreviations MUST only be added to the UnitsML approved acronym and abbreviation list after careful consideration for maximum understanding and reuse.

Once an acronym or abbreviation has been approved, it is essential to ensure, for semantic clarity and interoperability, that only the acronym or abbreviation shall be used. The use or inclusion of the spelled-out name for the acronym or abbreviation shall not be allowed.

[GNR6] The acronyms and abbreviations listed in Appendix A MUST always be used.

Generally speaking, the names for UnitsML XML constructs must be always singular. The only exceptions are words whose concept itself is plural.

[GNR7-UnitsML] UnitsML XML element and type names MUST be in singular form unless the concept itself is plural. The sole exceptions to this rules is for the naming of the "Conversions" and "RootUnits" elements.

Example:

Allowed - No singular concept available:

```
     <xsd:element name="GoodsQuantity" ...>
```

Not Allowed - Plural:

```
     <xsd:element name="ItemsQuantity" ...>
```

[GNR8-UnitsML] The UpperCamelCase (UCC) convention MUST be used for naming elements and complex types.

Example:

Allowed:

```
     <xsd:element name="ElementName" ...>
```

Not Allowed:

```
     <xsd:element name="elementName" ...>
     <xsd:element name="Elementname" ...>
```

[GNR9-UnitsML] The lowerCamelCase (LCC) convention MUST be used for naming attributes and simple types.

Example:

Allowed:

```
     <xsd:attribute name="attributeName" ...>
```
Not Allowed:

```
     <xsd:element name="AttributeName" ...>
```

[GNR10] Acronyms and abbreviations at the beginning of an attribute declaration MUST appear in all lower case. All other acronym and abbreviation usage in an attribute declaration MUST appear in upper case.

Example:

Allowed:

```
     <xsd:attribute name="siSystem" ...>
     <xsd:attribute name="prefixID" ...>
```

Not Allowed:

```
<xsd:attribute name="SISystem" ...>
```

[GNR11] Acronyms MUST appear in all upper case for all element declarations and
type definitions.

Example:

Allowed:

```
<xsd:element name="SISystem" ...>
<xsd:element name="PrefixID" ...>
```

Not Allowed:

```
<xsd:element name="SiSystem" ...>
<xsd:element name="PrefixId" ...>
```

2.7. Type Naming Rules

As declaration of the `name` attribute for `xsd:complexType` and `xsd:simpleType`
is not required by the W3C XML Schema standard, but since elements and types are
intended to be reusable, all types must be named. This permits other types to establish
elements that reference such types. The attribute `name` is not allowed for
`xsd:localComplexType` (local definition of types) therewith this rule implies global
type definitions.

[GTD1] All types MUST be named.

Example:

```
<xsd:complexType name="AccountType">
      <xsd:annotation>
      ... see annotation ...
      </xsd:annotation>
      <xsd:sequence>
      ... see element declaration ...
      </xsd:sequence>
```

```
          </xsd:complexType>
```

The purpose of UnitsML is to develop a schema for creating standardized and consistent XML documents containing a precise and unambiguous description of units. Permitting the use of the data type xsd:anyType would allow schema constraints to be circumvented. Not assigning types to attributes is equivalent to using the "simple urtype", i.e. xsd:anySimpleType. Thus attributes must have types assigned.

[GTD2-UnitsML] The xsd:anyType MUST NOT be used. Every attribute has an assigned type.

To distinguish type definitions for elements and attributes, type names for elements must appear in upper camel case and type names for attributes in lower camel case.

[UnitsML-GTD1] The UpperCamelCase (UCC) convention MUST be used for naming types used in elements.

Example:
```
          <xsd:complexType name="AccountType">
```

[UnitsML-GTD2] The lowerCamelCase (LCC) convention MUST be used for naming types used in attributes.

Example:
```
          <xsd:simpleType name="baseType">
```

2.8. Element Declarations

[ELD6] The code list `xsd:import` element MUST contain the namespace and schema location attributes.

Annotation:

Currently, no code lists are available for import. The rule ELD6 is considered for a future evolution of the UnitsML standard. Maybe this rule must be rewritten to restrict the use of `xsd:import` for incorporation of other schemas, such as MathML.

There is no need to declare empty elements.

[ELD7] Empty elements MUST not be declared.

The use of the `xsd:any` element permits the use of elements unknown at the time of schema creation. The usage may circumvent the precise and standardized description of units with the UnitsML schema and software cannot handle these unforeseen elements. The sole exception is for the SymbolType complex type, whose purpose is to allow usage of an undetermined, foreign markup language.

[ELD9-UnitsML] The `xsd:any` element MUST NOT be used. The single exception to this rule is within the SymbolType complex type which uses `xsd:any` on purpose.

2.9. Attribute Declaration

To provide XML-aware tools and XML validator with all referenced schemas, the `xsd:schemaLocation` attribute must contain a resolvable URL.

[ATD6] Each `xsd:schemaLocation` attribute declaration MUST contain a resolvable URL.

In general, the absence of an element in an XML schema does not have any particular

meaning. It may indicate that the information is unknown or not applicable or that the

element may be absent for some other reason. The XML schema specification does,

however, provide a feature - the `nillable` attribute - whereby an element may be

transferred with no content, but still use its attributes and thus carry semantic meaning.

To retain semantic clarity, the nillability feature of XSD will not be used.

> [ATD7] The `xsd` built in `nillable` attribute MUST NOT be used for any UnitsML
> declared element.

`xsd:Attribute` permits the use of attributes unknown at the time of schema creation.

The usage may circumvent the precise and standardized description of units with the

UnitsML schema and software cannot handle these unforeseen attributes.

> [ATD8] The `xsd:anyAttribute` MUST NOT be used.

2.10. General Schema Rules

This rule is useful to keep schemas as simple as possible. The unnecessary use of

complex types results in overly complex schemas.

> [GXS3] Built-in XSD Simple Types SHOULD be used wherever possible.

> [GXS4] All W3C XML Schema constructs in the UnitsML schema MUST contain the
> following namespace declaration on the XSD schema element:
> `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`

Substitution groups allow a global element to replace another global element in an XML

instance document without any further modifications to the schema. Substitution groups

disrupt the harmonization of element names.

> [GXS5] The `xsd:substitutionGroup` feature MUST NOT be used.

The `xsd:notation` attribute identifies a notation. Notation declarations affect all the

`<notation>` element information items in the `[children]`, if any, plus any included

or imported declarations. Per XML Schema Definition Part 2, "It is an error for NOTATION to be used directly in a schema. Only datatypes that are derived from NOTATION by specifying a value for enumeration can be used in a schema" [4]. The UnitsML schema model does not require or support the use of this feature.

[GXS7] `xsd:notation` MUST NOT be used.

The "`all`" compositor indicates that the elements declared within it can appear in any order. Because the UnitsML schema defines a special structure, it is important that element order be enforced in UnitsML instance documents or documents that incorporate UnitsML.

[GXS8] The `xsd:all` element MUST NOT be used.

The `xsd:include` feature provides a mechanism for bringing in schemas that reside in the same namespace. To avoid naming collisions with elements, types and attributes defined and declared in an incorporated schema, and avoid circular references this feature will not be used.

[GXS10] The `xsd:include` feature MUST NOT be used.

2.11. Instance Documents

[RED1] Every UBL instance document must use the global element defined as the root element in the schema as its root element.

Annotation:

"UBL has chosen a global element approach. In XSD, every global element is eligible to act as a root element in an instance document. Rule ELD1 (see below) requires the identification of a single global element in each UBL schema to be carried as the root element in the instance document. UBL business documents (UBL instances) must have a single root element as defined in the corresponding UBL XSD."

[ELD1] Each `UBL:DocumentSchema` MUST identify one and only one global element

declaration that defines the document

`ccts:AggregateBusinessInformationEntity` being conveyed in the Schema expression. That global element MUST include an `xsd:annotation` child element which MUST further contain an `xsd:documentation` child element that declares "*This element MUST be conveyed as the root element in any instance document based on this Schema expression.*"

There are some problems with the identification of the root element and the enforcement with standard tools (XML validators). We should think about the approach of global element declarations. Additionally this is intended for document centric schemas only and not for data centric schemas such as the UnitsML.

Instance documents that do not validate against the UnitsML schema are not considered as UnitsML markup.

[IND1] All UnitsML instance documents MUST validate to the UnitsML schema.

XML supports a wide variety of character encodings. Processors must understand which character encoding is employed in each XML document. XML 1.0 supports a default

value of UTF-8 for character encoding, but best practice is to always identify the character encoding being employed.

[IND2] All UnitsML instance documents MUST always identify their character encoding with the XML declaration.

UnitsML, as an upcoming OASIS TC, is obligated to conform to agreements OASIS has entered into. OASIS is a liaison member of the ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group (MOUMG). Resolution 01/08 (MOU/MG01n83) requires the use of UTF-8.

[IND3] In conformance with ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group (MOUMG) Resolution 01/08 (MOU/MG01n83) as agreed to by OASIS, UnitsML XML SHOULD be expressed using UTF-8.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

3. Acknowledgements

In alphabetical order:

- Jens Bakoczy (NIST, BFRL, Gaithersburg, MD, U.S.A.)

- Mark Carlisle (NIST, MEL/SIMA, Gaithersburg, MD, U.S.A.)

- Ismet Celebi (NIST, Physics, Gaithersburg, MD, U.S.A.)

- Mark Crawford (LMI)

- Bob Dragoset (NIST, Physics, Gaithersburg, MD, U.S.A.)

- Simon Frechette (NIST, MEL/SIMA, Gaithersburg, MD, U.S.A.)

- Gary Kramer (NIST, CSTL, Gaithersburg, MD, U.S.A.)

- Peter Linstrom (NIST, CSTL, Gaithersburg, MD, U.S.A.)

- Karen Olsen (NIST, Physics, Gaithersburg, MD, U.S.A.)

- Kent Reed (NIST, BFRL, Gaithersburg, MD, U.S.A.)

- Martin Weber (NIST, Physics, Gaithersburg, MD, U.S.A)

4. References

1. **NIST UnitsML working group**

    http://unitsml.nist.gov

2. **OASIS Universal Business Language (UBL)**

    http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl

3. **"Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, IETF**

    http://www.ietf.org/rfc/rfc2396.txt

4. **"A URN Namespace for OASIS", RFC 3121, IETF**

    http://www.ietf.org/rfc/rfc3121.txt

5. **U.S. Department of the Navy NDR Project**

    http://xml.coverpages.org/DON-XML-NDR20050127-33942.pdf

6. **W3C XML Schema Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes**

    http://www.w3.org/XML/Schema

    http://www.w3.org/TR/xmlschema-1/

    http://www.w3.org/TR/xmlschema-2/

5. Appendix A: Allowed abbreviations for UnitsML

ASCII American Standard Code for Information Interchange

ID   Identifier

lang   language

ML   Markup language

~~URI   Uniform Resource Identifier~~

~~URL   Uniform Resource Locator~~

SI   The International System of Units

> http://www.bipm.fr/en/si/
>
> http://physics.nist.gov/cuu/Units/

~~lang   language~~

UnitsML   Units Markup Language

URI   Uniform Resource Identifier

URL   Uniform Resource Locator

WSDL Web Service Description Language