Draft Paper: Position Paper on the localization rules of UBL 1.0

Draft 1, 7 Oct. 2005

Document identifier:

Bryan's document as per the title

Location:

Author:

Bryan Rasmussen <brs@itst.dk>

Abstract:

This position paper brings some points to discussion regarding Localization.

Status:

Table of Contents

Introduction

The purpose of the Localization project is generally understood to be twofold:

1. Customizing UBL to meet local requirements, and
2. Achieving (1) whilst being “conformant” to UBL

As such there is no actual requirement that Localization should take place under the auspices of XML Schema validation of local instances and that the XML Schemas used have some relationship with the UBL XML Schemas. Nonetheless I think in any practical discussion this is what the subject will boil down to, and I think this is somewhat unfortunate because in my experience I do not find XML Schema flexible enough to make ‘reuse’ of any Schema or set of Schemas, of sufficient complexity, a practical matter. If I had to make an ERP system that used a specialized format of UBL I would copy the schemas and make the changes by hand at the locations where I wanted my rules to diverge from the normal UBL rules (assuming of course that I decided to use XML Schemas in my validation process), and in that context trying to worry about the reusability of a schema is nonsensical.

My example of an ERP system usage of UBL is of course not on the same level as a Localization level that supposes a reuse scheme for a standardization effort on the National or even International Level. The ERP system would derive obvious benefits from rewriting the actual schemas in the following ways:

1. The number of schemas they would have to manage would be reduced.
2. Most probably the physical size of the schema files would be reduced.
3. If the system was based on a particular XML Schema processor the rewritten schemas could be structured in such a way and use facilities of XML Schema that would work better in a particular processor.

These benefits are not present on the level of another standardization body building on top of the standardization efforts of the UBL TC.
Thus the question of Localization becomes not just how to best facilitate Localization using XML Schemas but what can be done to aid this super-standardization.

The concept of super-standardization

As defined in the introduction the concept of super-standardization is one where a standardization body builds a standard on top of the standardization efforts of another one, we should also note that it relates to a specialization of the standard to either the needs of a particular domain (as is seen with the case of the Small-Business Subset) or the needs of a particular region, which could relate to the needs of the EU for adapting UBL messages or the needs of the Danish government for doing the same.

It is the super-standardization to meet the needs of a region that we would refer to as Localization.

As a general rule I think the requirements for making a specialized standard by domain and by region would be similar with the following exceptions under region:

1. Restrictions of human languages allowed in documents
2. Restrictions of currencies allowed in documents

As such I am not sure if it would be especially useful to focus on the needs for specialization by region as opposed to specialization as a whole and the super-standardization process that will produce a specialization.

Experiences of Danish super-standardization

The Experiences of Danish super-standardization may turn out to be somewhat unique in relation to most super-standardization in that it was legally mandated, with a very short time line moving from the point of mandate to the point of implementation.

It should be noted that there were a number of difficulties for various organizations to bring their systems into compliance with the legal mandate, however these difficulties cannot be solely blamed on the short time from mandate to implementation given that a number of organizations seemed to adopt a strategy of wait until the deadline and then buy something quickly to handle the problem. It is not unduly cynical to assume the same strategy will occur in other situations.

The reason for the legal mandate for moving to UBL in Denmark was to derive the monetary benefits of doing so in savings of Governmental Invoice handling. As such the standardization in Denmark was focussed on making the UBL messages sent and received suitable for use with the main ERP systems used in the Danish Government at various levels.

On the position of independent local versions of the schemas

The idea that the best way to provide interoperability is not to worry about extending/reusing the schemas I will term the position of ‘independent local versions’.

In an email from Jon Bosak to the list http://www.oasis-open.org/apps/org/workgroup/ubl/email/archives/200508/msg00042.html on the subject of code-lists (a subset of the whole localization problem) he says:

“There appear to be three ways to accomplish modifications to
UBL schemas without changing the namespaces”
And enumerates the ways with the first one being:

“Users simply modify the file containing the code list while
leaving everything else alone. This method is being used
successfully in Denmark. Obviously we cannot prevent users
from doing this, and given a proper notification procedure,

it seems to work pretty well."

Which seems to acknowledge this position as an allowed one. I will not focus further on this position given that it is basically an argument that independence of local and international versions is to be preferred, and that this argument can really only be based on the problems associated with the more specific positions of integration between the local and international versions – said integration being known as reuse.

It should however be noted that Mr. Bosak makes reference to "a proper notification procedure", I am not sure that there should not be defined a proper notification procedure for Localization which could be used to provide interconnection between versions in various projects. As to what such a notification procedure would require, it is outside the scope of this document.

On the use of XML Schema Constructs to enable easier localization

Whatever other considerations that are involved with Localization discussed above we will still need to focus on easing the use of XML Schemas in the Localization process. Generally it has been argued that the easiest way to do this is to reuse the schemas and use XML Schemas methods for overwriting various classes and types as needed. I have not found the generally used methods of XML Schema reuse especially useful, preferring instead to use a higher-level language or a hand-rolled domain specific language to generate the Schemas whole. However there are three commonly used XML Schema Constructs to enable easier reuse of Schemas than is allowed by the common methods of working with XML Schemas and these are Substitution Groups, Redefines, and use of the xsd:any construct.

Substitution Groups were discussed as a way to enable easier reuse of the schemas for localization purposes as applicable specifically to reuse of codelists in the UBL TC in a discussion that has been rather wide-ranging and long running.

I discussed this on the XML-Schema dev list in the following email
http://lists.w3.org/Archives/Public/xmlschema-dev/2005Jul/0005
I was hoping to get a good overview of support of Substitution Groups among the various processors, although as noted I was also interested in confirming that my experiences with xsd:redefine was also shared by others as redefine was at that point being suggested as a possible substitution mechanism instead of substitution groups.

Responses can be read in the list, to sum up I would say the following:

Substitution Groups are generally well supported, with the exception of a couple of the major processors, inclusive .Net ValidatingReader version 1.1 (my own personal experience in this case).

Redefine is not commonly well supported and should not be used (personal experience and group consensus).

Given that Substitution Groups have some problems with support I would suppose that relying on them could possibly make development of UBL applications more expensive.

On a personal note the syntax of Substitution Groups are difficult for me to understand and I have problems understanding and tracking their usage when I encounter them. Use of xsd:any and xsd:redefine are in contrast not difficult at all.

The Danish Experience with use of xsd:any

The use of xsd:any by the Danish super-standardization of UBL 0.7 was haphazard, based on a meeting with the Danish Ministry of the Economy in which they told me that they had been using note fields in an invoice to pass around escaped XML fragments needed by their applications for tracking of the XML document through the system, reading in those escaped XML fragments, unescaping them, loading them in as an XML Dom and so forth as needed. The reason why the format could not just be extended with custom markup at relevant places was that validation of an Invoice did not just take place at the entry into the system and the exit, but rather all through the system at various points the number and purpose of which I have absolutely no knowledge about.

Given this, and given that the system used by the Ministry of the Economy was the main one that all governmental invoices would have to pass through and thus was the system through which the most savings could be expected I suggested the simple expedient of adding an element ExtensibleContent that would require one child element in any namespace not among the Danish UBL namespaces, and that the validation of this child element should be set to skip validation. Rules were specified that any information found in there that was not understood by an application should be ignored, the ExtensibleContent element itself was considered to be an agreed up element between trading partners, if anyone wanted to use it to exchange information between trading partners not currently specified in the UBL message itself.

That, as far as I was concerned, was the whole of it – although I did consider making some demonstrations of passing around xhtml displays of the message in the area.

As it turned out however the ExtensibleContent element later proved to be very useful, so useful in fact that if we had not had it this document might not exist, because the Danish UBL super-standardization effort might have run into major problems at lacking it, been cancelled, and hence not been available to produce this document.

As noted earlier one of the defining characteristics of the Danish super-standardization effort that may prove to be an uncommon one is that it was legally mandated. Basically all organizations had to invoice the Government using the Danish version of UBL. However after the date of the implementation was passed we found out from various utility organizations that our law mandating UBL was in conflict with a law applying to their industry mandating what information had to be sent with an Invoice.
Obviously such a situation has the potential of becoming politically untenable. Luckily we had a solution provided by our use of xsd:any, which allowed the utility industry to identify additions to the legally mandated format without breaking the law requiring invoicing information to be sent in that format.

I am not certain that any super-standardization work on a regional level will have the resources to match all the requirements of specific domains operating within their region for all the possible document formats specified, of course if legal mandates are not in place forcing the use of UBL those unable to comply can always escape into legacy paper formats.

Use of xsd:any

Given the Danish experience with the use of xsd:any we are very committed to allow extension of the message as the necessity arises.
This was brought to the attention of the UBL TC but the response was largely negative. I believe this split was based firstly on a lack of understanding of the problem on a practical level, and a lack of knowledge about the possible uses.

The description of the problem on a practical level was given in the Danish experience with xsd:any.

I think one of the reasons for the largely negative reaction to the use of xsd:any was the perception that validation could not be specified for the namespace that was used within the any area. This is dependent on the choice of what processing model is used for the elements in the other namespace.

In the Danish implementation we specified that the processing for elements in the other namespace would be skipped, however it is possible to specify that the processing for elements in the other namespace must be lax or even strict. The following example schemas and instance show one possible usage scenario of such a structure:

Example Schema 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://test.org" xmlns:b="http://test.org"
xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
                <element name="a" type="b:aType">

                </element>
                <complexType name="aType">
                <sequence>
                        <any namespace="##other" minOccurs="1" maxOccurs="1"/>
                </sequence>
                </complexType>
</schema>
```

Example Schema 2:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://test.com" xmlns:b2="http://test.com"
xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
                <element name="a2" type="string">

                </element>
</schema>
```

Example XML Instance

```
<?xml version="1.0" encoding="utf-8"?>
<b:a xmlns:b="http://test.org" xmlns:b2="http://test.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://test.com test2.xsd"
>
<b2:a2>hello</b2:a2>
</b:a>
```

Running the above example instance against XSV, Microsoft's SchemaCache, and .Net validatingReader version 1.1 with the top schema for the namespace http://test.org defined programmatically raises no errors. Of course this only works if one specifies that second schema in the xsi:schemaLocation, which I believe is too fragile for standard usage. However it is possible in the greatest number of libraries for Schema Validation to provide this functionality programmatically, i.e. without any recourse to xsi:schemaLocation.

The foregoing is not a recommendation for how to proceed but rather is presented as a demonstration that there are still numerous possibilities to restriction of the xsd:any, i.e. that it does not need to be just 'anything' is allowed in an any.

Furthermore while it does open up the structure of the extensible area we can still place attributes on the area itself, useful for determining the meaning of the contained markup, its relation to the actual UBL document, and the issuing organization of the extended format. The possibility of adding attributes to the extensible area was not addressed by the Danish committee, so as to avoid creating more discussion about an element that was in the first draft considered basically for application specific extension.

Possible usage scenarios of xsd:any

Aside from the usages scenarios described where the Danish version of UBL was concerned other possible usage scenarios of xsd:any would include:

1. Uses where one mainly wanted to extend and not restrict UBL.
2. In cases where restriction was needed the normal restrictions could be used, or a lazy man's version of specifying default values for ignored elements that would be replaced by elements within the #any namespace within the container element.

3. A super-standardization work could specify extension of UBL document types with various other technologies, one example would be extension with XLink or RDF, in such a manner that namespaces in the extended areas could be tracked back to the structures in the UBL namespace with which they were actually related.

Why not extend at the envelope level?

In discussion with the UBL TC one argument against usage of xsd:any was that the proper place for any extension of the message of such a nature should be done at the envelope level, I am against this for the following reasons:

1. This implies that any group creating a super-standardization of UBL should also specify protocol requirements. Which was not done in the Danish practice and which it had been attempted would have been politically impractical to complete.
2. In Denmark, which may be the same situation elsewhere, there are very strict rules regarding archiving of official documents. Considering such extension at the envelope level I could definitely envision a situation where not just the individual messages contained within an envelope should be archived, but also the envelope itself.