Position Paper: Local vs global

Proposal 01, 28 May 2002

Document identifier:

p-desre-localvsglobal-01 (Word)

Location:

http://www.oasis-open.org/committees/ubl/ndrsc/pos

Authors:

Fabrice Desré, France Télécom <fabrice.desre@francetelecom.com>

Abstract:

This position paper outlines several options related to the use of global elements when designing the UBL schema library.

Status:

This is V01 of the local vs. global position paper intended for consideration by the OASIS UBL Naming and Design Rules subcommittee and other interested parties. It is still incomplete; however, parts of the recommendations made here have already been incorporated into the *Universal Business Language Naming and Design Rules Specification*.

If you are on the ubl-ndrsc@lists.oasis-open.org list for subcommittee members, send comments there. If you are not on that list, subscribe to the ubl-comment@lists.oasis-open.org list and send comments there. To subscribe, send an email message to ubl-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

Table of Contents

1 Introduction

The UBL schema library consists of a collection of data types, which are used either in the UBL schemas themselves or in third parties schemas ("extended", "customized" schemas).

The Naming and Design Rules subcommittee (NDRSC) must make some decisions to specify how valid UBL schemas can be built. Among these decision, the NDRSC must specify if the UBL types should be backed up by global elements, and whether these elements should be namespace-qualified or not.

The original decision on this matter (see wd-ublndrsc-ndrdoc-08.doc) is to have globally declared qualified root elements, and locally declared unqualified sub elements.

2 Requirements

The following requirements were identified by the NDRSC :

- **Compatibility with previous NDR decisions**, e.g. element naming conventions (see wd-ublndrsc-ndrdoc-08.doc).
- **Modeling flexibility** : content models must be able to include two elements of the same type.

  The local vs. global decision must not preclude the design of complex types that carry several fields of the same type.
- **Context rules friendliness**

  The ability to use normal mechanisms of the context methodology without adding burden. Several approaches exist for context methodology : Paella and TAAT.
- **XSLT/XPath friendliness**

  The local vs. global decision has potential impacts on what the instances look like, and what the resulting XML data model is. We have to come with a non intrusive solution allowing the use similar of XPath expressions for standard and customized UBL instances.
- **Ease of processing**

  We must take into account that some applications need to parse documents fragments easily, without having the whole document context knowledge.

3 XSD design

In this section we will outline some XSD design considerations useful for the following sections of this document.

3.1 Global elements

A global element can be built in two ways :

- Using a standalone element

```
<xs:element name="info">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="age" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- Using a type reference

```
<xs:complexType name="infoType">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="age" type="xs:int"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="info" type="infoType"/>
```

Each of these schemas be used to validate the same instance :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<info>
<name>Fabrice Desré</name>
<age>29</age>
<info>
```

3.2 Namespace interaction

The previous example uses unqualified elements. However we will use namespaces in UBL for several purposes (modularity, versioning, extensions), so we must be aware of the consequences.

When writing schemas, each module (i.e. file) end up with its own namespace target. A common use case is then to reuse these schemas by importing them (using <xsd:import/>). The behavior of such imported components can be described that way :

- Imported types don't preserve the namespace of their original schema. The newly declared elements belong to the new namespace.
- Referenced global elements preserve the namespace of their original schema.

Here are two sample schemas to exhibit these differences :

File : localvsglobal.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.oasis-open.org/ubl"
          xmlns:ubl="http://www.oasis-open.org/ubl"
          elementFormDefault="qualified">

<xs:element name="info">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="age" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:complexType name="infoType">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="age" type="xs:int"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="infoRef" type="ubl:infoType"/>

</xs:schema>
```

File : customized.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.foo.org/"
          xmlns:ubl="http://www.oasis-open.org/ubl"
          xmlns:foo="http://www.foo.org/"
          elementFormDefault="qualified">

<xs:import namespace="http://www.oasis-open.org/ubl"
           schemaLocation="localvsglobal.xsd"/>

<xs:complexType name="myFirstInfo">
  <xs:sequence>
    <xs:element name="info" type="ubl:infoType"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

```
<xs:element name="info1" type="foo:myFirstInfo"/>

<xs:complexType name="mySecondInfo">
  <xs:sequence>
    <xs:element ref="ubl:info"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="info2" type="foo:mySecondInfo"/>

</xs:schema>
```

And two valid instances :

```
<?xml version="1.0" encoding="UTF-8" ?>
<foo:info1 xmlns:ubl="http://www.oasis-open.org/ubl"
          xmlns:foo="http://www.foo.org/">
  <foo:info>
    <foo:name>Fabrice</foo:name>
    <foo:age>29</foo:age>
  </foo:info>
  <foo:country>France</foo:country>
</foo:info1>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<foo:info2 xmlns:ubl="http://www.oasis-open.org/ubl"
          xmlns:foo="http://www.foo.org/">
  <ubl:info>
    <ubl:name>Fabrice</ubl:name>
    <ubl:age>29</ubl:age>
  </ubl:info>
  <foo:country>France</foo:country>
</foo:info2>
```

We can see that in the first instance (info1), the UBL type "info" has been borrowed and is now referred as "foo:info". On the other side, the second instance preserve the real origin of the type.

4 Options

I will consider two options:

- The current UBL recommendation
- An alternative one, trying to better fulfill our requirements.

For each one, a sample UBL schema will be given, along with a customized one and the relevant instances.

4.1 Current UBL recommendation

The current UBL recommendation dictates to :

- Use qualified elements for root elements in the standard UBL library.
- Use unqualified elements for every other elements in the standard UBL library.
- Extension writers must use qualified elements.

4.1.1 Schema definitions

An hypothetical UBL order schema :

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.oasis-open.org/ubl"
          xmlns:ubl="http://www.oasis-open.org/ubl"
          elementFormDefault="unqualified">

<xs:complexType name="orderType">
  <xs:sequence>
    <xs:element name="header" type="ubl:headerType"/>
    <xs:element name="item" type="ubl:itemType"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="headerType">
  <xs:sequence>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="to" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="itemType">
  <xs:sequence>
    <xs:element name="desc" type="xs:string"/>
    <xs:element name="price" type="xs:double"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="orderDoc" type="ubl:orderType"
form="qualified"/>

</xs:schema>
```

An extension/reuse of this schema to build a custom order :

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://example.org/foo"
          xmlns:ubl="http://www.oasis-open.org/ubl"
          xmlns:foo="http://example.org/foo"
          elementFormDefault="qualified">

<xs:import namespace="http://www.oasis-open.org/ubl"
           schemaLocation="currentUBL.xsd"/>

<xs:complexType name="fooHeaderType">
  <xs:sequence>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="date" type="xs:dateTime"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="fooOrderDoc">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="header" type="foo:fooHeaderType"/>
      <xs:element name="item" type="ubl:itemType"
maxOccurs="unbounded"/>
      <xs:element name="desc" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

4.1.2 Instances

Here are the basic and the customized instances :

```
<?xml version="1.0" encoding="UTF-8" ?>
<ubl:orderDoc xmlns:ubl="http://www.oasis-open.org/ubl">
  <header>
    <from>Fabrice</from>
    <to>NDRSC</to>
  </header>
  <item>
    <desc>first item</desc>
    <price>10</price>
  </item>
  <item>
    <desc>second item</desc>
    <price>20</price>
  </item>
```

```
</ubl:orderDoc>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<foo:fooOrderDoc xmlns:ubl="http://www.oasis-open.org/ubl"
          xmlns:foo="http://www.foo.org/">
  <foo:header>
    <foo:from>Fabrice</foo:from>
    <foo:to>NDRSC</foo:to>
  </foo:header>
  <foo:item>
    <foo:desc>first item</foo:desc>
    <foo:price>10</foo:price>
  </foo:item>
  <foo:item>
    <foo:desc>second item</foo:desc>
    <foo:price>20</foo:price>
  </foo:item>
  <foo:desc>A sample order</foo:desc>
</foo:fooOrderDoc>
```

4.1.3 Requirements analysis

Compatibility with previous NDR decisions	+++ Implied by this option
Modeling flexibility	+++ Elements use type reference inside complex content models, thus allowing
Context rules friendliness	??
XSLT/XPath friendliness	--- Consider the Xpath expression needed to access an item's price. Since the item type is simply reused, we can except similar solutions for both : • Within the standard instance : //item/price • Within the customized instance : //foo:item/foo:price Worst, an XSLT template matching rule to display an item would be : • <xsl:template match="item"> for the standard library. • <xsl:template match="foo:item"> for the customized library. This implies that standard stylesheets can't display

	customized documents in a backward compatible way (they could by using the local-name() Xpath function, but this means that we don't care at all about namespaces !).
Ease of processing	-- There is no way looking at the instance to know that <item/> and <foo:item/> have indeed the same content model. This implies that fragment processing application must be aware of this (how? Using the standard dictionary and an augmented one ?).

4.2 Globally qualified solution

This is an attempt to provide a design pattern that will fulfill at best our requirements. The previous analysis showed us that the main drawbacks of the current solution are in the "namespace qualilfying" area.

The so called "globally qualified" solution can be viewed as an attempt to :

- Localize type definitions inside a namespace to allow type reuse.
- Create aliases to these types to preserve original namespace in customized schemas.

4.2.1 Schema definitions

An hypothetical UBL order schema :

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.oasis-open.org/ubl"
          xmlns:ubl="http://www.oasis-open.org/ubl"
          elementFormDefault="qualified">

<xs:complexType name="orderType">
  <xs:sequence>
    <xs:element name="header" type="ubl:headerType"/>
    <xs:element name="item" type="ubl:itemType"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="headerType">
  <xs:sequence>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="to" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="itemType">
```

```
  <xs:sequence>
    <xs:element name="desc" type="xs:string"/>
    <xs:element name="price" type="xs:double"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="orderDoc" type="ubl:orderType"/>

<xs:element name="headerTypeRef" ref="ubl:headerType"/>
<xs:element name="itemTypeRef" ref="ubl:itemType"/>
</xs:schema>
```

An extension/reuse of this schema to build a custom order :

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://example.org/foo"
          xmlns:ubl="http://www.oasis-open.org/ubl"
          xmlns:foo="http://example.org/foo"
          elementFormDefault="qualified">

<xs:import namespace="http://www.oasis-open.org/ubl"
           schemaLocation="currentUBL.xsd"/>

<xs:complexType name="fooHeaderType">
  <xs:sequence>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="date" type="xs:dateTime"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="fooOrderDoc">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ubl:headerTypeRef"/>
      <xs:element ref=" ubl:itemTypeRef" maxOccurs="unbounded"/>
      <xs:element name="desc" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

4.2.2 Instances

Here are the basic and the customized instances :

```
<?xml version="1.0" encoding="UTF-8" ?>
<orderDoc xmlns ="http://www.oasis-open.org/ubl">
  <header>
```

```
    <from>Fabrice</from>
    <to>NDRSC</to>
  </header>
  <item>
    <desc>first item</desc>
    <price>10</price>
  </item>
  <item>
    <desc>second item</desc>
    <price>20</price>
  </item>
</orderDoc>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<foo:fooOrderDoc xmlns:ubl="http://www.oasis-open.org/ubl"
          xmlns:foo="http://www.foo.org/">
  <foo:header>
    <foo:from>Fabrice</foo:from>
    <foo:to>NDRSC</foo:to>
  </foo:header>
  <ubl:itemTypeRef>
    <ubl:desc>first item</ubl:desc>
    <ubl:price>10</ubl:price>
  </ubl:itemTypeRef>
  <ubl:itemTypeRef>
    <ubl:desc>second item</ubl:desc>
    <ubl:price>20</ubl:price>
  </ubl:itemTypeRef>
  <foo:desc>A sample order</foo:desc>
</foo:fooOrderDoc>
```

4.2.3 Requirements analysis

Compatibility with previous NDR decisions	++ We'll probably need to write a recommendation for the naming of elements that are type aliases.
Modeling flexibility	-- In each module, we can reuse a type several times in a content model. But a customized schemas can't reference several times an imported type.
Context rules friendliness	??
XSLT/XPath friendliness	+++ Consider the Xpath expression needed to access an item's

	price. Since the item type is simply reused, we can except similar solutions for both : • Within the standard instance : //ubl:item/ubl:price • Within the customized instance : //ubl:itemTypeRef/ubl:price In the same vein an XSLT template matching rule to display an item would be : • <xsl:template match="ubl:item|ubl:itemTypeRef"> for both. This shows that we don't have to update the standard stylesheets to provide a default display to customized instances.
Ease of processing	+++ The application only needs to know about the UBL standard element name and the aliased name that can possibly appear in a customized instance.

410

411 # 5 Issues

412 The pending issues so far are :

413 • Modeling flexibility with the globally qualified solution

414 • Context rules friendliness : we need to know more about TAAT formalism to analyse this.

Appendix A. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.