Universal Business Language (UBL) Position Paper:

Using List Containers

Authors:

Tim McGrath (tmcgrath@portcomm.com.au)

Stephen Green (stephen_green@bristol-city.gov.uk)

Chin Chee-Kai (cheekai@SoftML.Net)

Bill Meadows (Bill.Meadows@Sun.COM)

Anthony B. Coates (abcoates@londonmarketsystems.com)

Date: August 30, 2003

Filename: position-lcsc-listcontainers-02.doc

Table of Contents

1Summary 3
2Problem Description 4
3Case when not using List Containers 6
3.1Models 6
3.2Schemas 7
3.3Instances 7
4Case when using Selected List Containers 9
4.1Models 9
4.2Schemas 10
4.3Instances 10
5Case when using all List Containers 11
5.1Models 11
5.2Schemas 11
5.3Instances 12
6Findings 13
6.1Benefits of List Containers 13
6.2Problems with using List Containers 14
7Conclusion 17

1 Summary

This document represents the outcome of a test implementation of the NDR rules concerning List Containers.

Due to the overall level of concern about this matter it was felt that a real world example would be necessary to identify the strengths and weaknesses of implementing these rules and allow an objective debate on their value.

Three versions of the current UBL Library were developed and implemented in conceptual model, XSD Schema and XML instances. These were:

a. The current UBL Library with no list container structures

b. The current UBL Library with suitable list container structures identified by the LC team

c. The current UBL Library with list container structures for all BIEs with potentially multiple occurrences (i.e a cardinality of either 0..n or 1..n)

From this exercise we have been unable to demonstrate any pronounced architectural benefits in using List Containers, either in processing performance or readability. Furthermore, even if some easily recognized value could be identified for using list containers, these would need to outweigh the difficulty of their consistent implementation into the UBL library.

We therefore suggest that Rule 116 be reconsidered by the UBL NDR team.

2 Problem Description

The UBL Naming and Design Rules rules number 116 states:

```
All elements with a cardinality of 1..n, (and lack a qualifying
structure)¹ MUST be contained by a list container named "(name
of repeating element)List", which has a cardinality of 1..1.
```

Importantly, these rules exist to satisfy syntactical features of the XML/XSD environment and do not impact on the conceptual models of the Library itself. These conceptual models identify and define the semantic containers (ABIEs) of the UBL Library. By definition, list containers are XML-wrappers used for encapsulating one or more instances of the same structures.

An amendment has subsequently been proposed for Rule 116 to deal with the case of 0..n occurrences (and to make the rule generic). This is...

```
Suppose that the <Thing> element has cardinality M..N. Then
1. If N <= 1 (i.e. <Thing> is 0..1 or 1..1), then <Thing>
does not have a container.
2. If N >= 2 (e.g. 0..2, 1..2, 2..2, 0..3, 1..3, ...) then
<Thing> has a <ThingList> container.
3. If M = 0, the cardinality of <Thing> inside <ThingList>
is 1..N, and the cardinality of <ThingList> is 0..1.
4. If M >= 1, the cardinality of <Thing> inside <ThingList>
is M..N, and the cardinality of <ThingList> is 1..1.
```

The rationales for having this rule are:

a. list container elements foster more readable schemas and instances.

b. list container elements improve document processing performance (such as stylesheet processing)

and

c. other XML vocabularies use these constructs

However, there are also potential overheads associated with the use of list containers:

a. Additional levels of complexity in the Schemas (e.g. XPath of elements)

b. Increase in size of the UBL Schemas

and

c. A separation/mismatch between the structures in the conceptual UBL models (ie. the spreadsheets and class diagrams) and the normative UBL Schemas. For example, structures that are optional in the models appear mandatory in the schemas.

In addition, some members of the UBL TC felt that other, as yet unknown, side effects may also be felt from introducing these constructs. For example, the requirement to provide additional business rules to determine the identification of candidates for list containers.

As such, there was concern that the benefit from using list containers may be less than the cost. This paper documents the test cases built to prove or disprove these benefits and costs.

---

[1] We were unable to get clarification on what a 'qualifying structure' is that may relax this rule. We have had to make the assumption that we have no such structures in the current library.

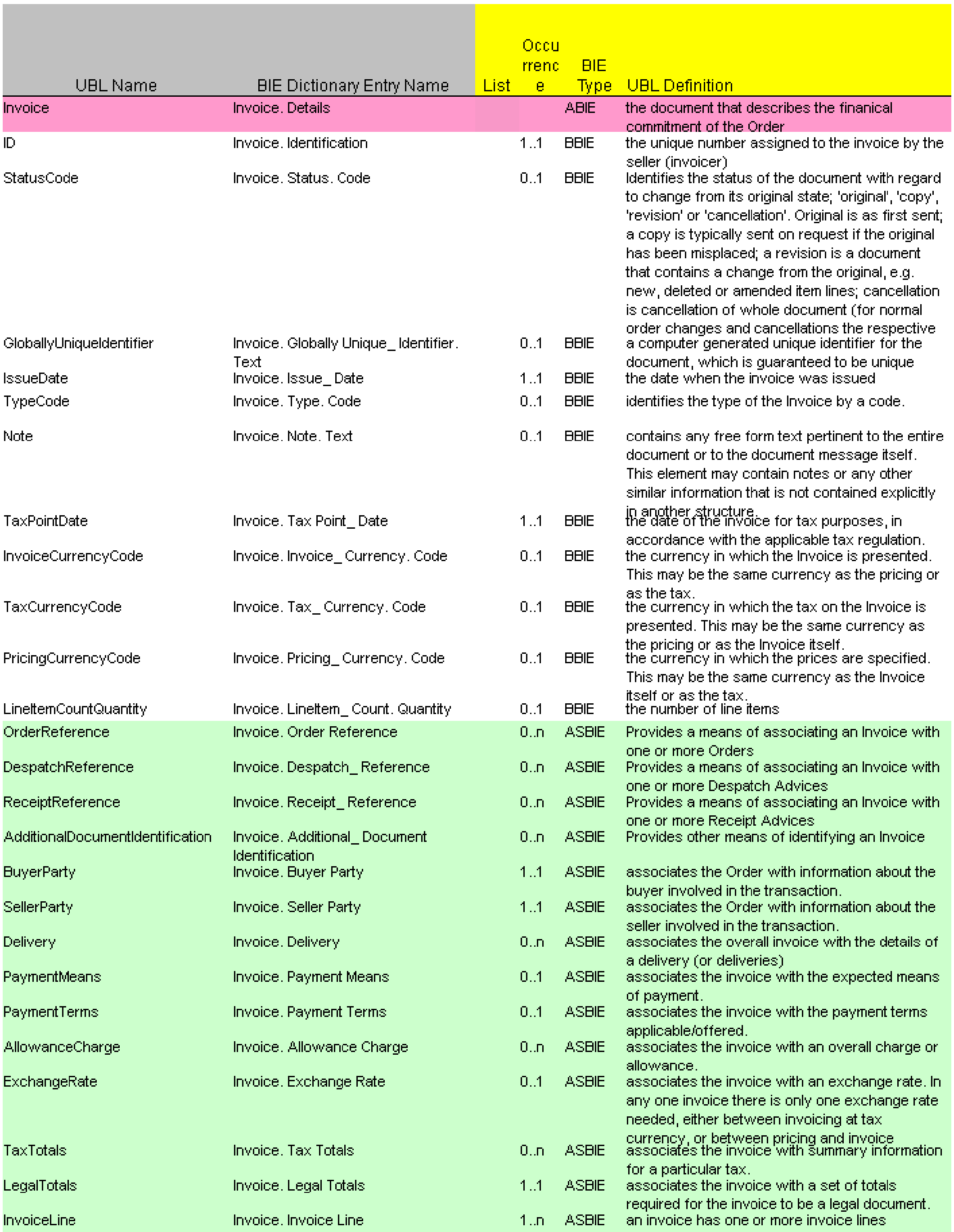
The entire schemas and example instance documents are available as attachments to this document, but for simplicity we have decided to use a representative fragment for our comparisons. The TaxTotals ABIE contains demonstrates the differences between the three approaches. It occurs in a 0..n association with Invoice and contains another 0..n association (with TaxSubTotals).

There is a note of caution when using a 'small' example such as TaxTotals. One of the issues of concern here is scalability. Performance and readability issues impact differently depending on the scale or complexity of documents. Therefore, we recommend the reader take time to study other structures in the attached document beyond these examples.

3 Case when not using List Containers

3.1 Models

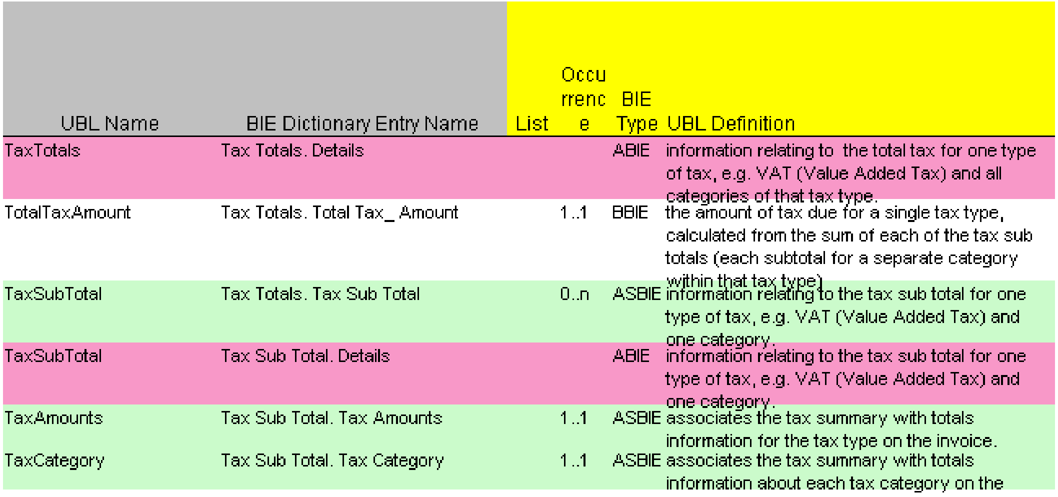
The following is a section of the current UBL model describing the Invoice document. There are no indicators in the “List” column to denote any list containers are required.



UBL Name	BIE Dictionary Entry Name	List	Occurrence	BIE Type	UBL Definition
Invoice	Invoice. Details			ABIE	the document that describes the finanical commitment of the Order
ID	Invoice. Identification		1..1	BBIE	the unique number assigned to the invoice by the seller (invoicer)
StatusCode	Invoice. Status. Code		0..1	BBIE	Identifies the status of the document with regard to change from its original state; 'original', 'copy', 'revision' or 'cancellation'. Original is as first sent; a copy is typically sent on request if the original has been misplaced; a revision is a document that contains a change from the original, e.g. new, deleted or amended item lines; cancellation is cancellation of whole document (for normal order changes and cancellations the respective
GloballyUniqueIdentifier	Invoice. Globally Unique_ Identifier. Text		0..1	BBIE	a computer generated unique identifier for the document, which is guaranteed to be unique
IssueDate	Invoice. Issue_ Date		1..1	BBIE	the date when the invoice was issued
TypeCode	Invoice. Type. Code		0..1	BBIE	identifies the type of the Invoice by a code.
Note	Invoice. Note. Text		0..1	BBIE	contains any free form text pertinent to the entire document or to the document message itself. This element may contain notes or any other similar information that is not contained explicitly in another structure.
TaxPointDate	Invoice. Tax Point_ Date		1..1	BBIE	the date of the invoice for tax purposes, in accordance with the applicable tax regulation.
InvoiceCurrencyCode	Invoice. Invoice_ Currency. Code		0..1	BBIE	the currency in which the Invoice is presented. This may be the same currency as the pricing or as the tax.
TaxCurrencyCode	Invoice. Tax_ Currency. Code		0..1	BBIE	the currency in which the tax on the Invoice is presented. This may be the same currency as the pricing or as the Invoice itself.
PricingCurrencyCode	Invoice. Pricing_ Currency. Code		0..1	BBIE	the currency in which the prices are specified. This may be the same currency as the Invoice itself or as the tax.
LineItemCountQuantity	Invoice. LineItem_ Count. Quantity		0..1	BBIE	the number of line items
OrderReference	Invoice. Order Reference		0..n	ASBIE	Provides a means of associating an Invoice with one or more Orders
DespatchReference	Invoice. Despatch_ Reference		0..n	ASBIE	Provides a means of associating an Invoice with one or more Despatch Advices
ReceiptReference	Invoice. Receipt_ Reference		0..n	ASBIE	Provides a means of associating an Invoice with one or more Receipt Advices
AdditionalDocumentIdentification	Invoice. Additional_ Document Identification		0..n	ASBIE	Provides other means of identifying an Invoice
BuyerParty	Invoice. Buyer Party		1..1	ASBIE	associates the Order with information about the buyer involved in the transaction.
SellerParty	Invoice. Seller Party		1..1	ASBIE	associates the Order with information about the seller involved in the transaction.
Delivery	Invoice. Delivery		0..n	ASBIE	associates the overall invoice with the details of a delivery (or deliveries)
PaymentMeans	Invoice. Payment Means		0..1	ASBIE	associates the invoice with the expected means of payment.
PaymentTerms	Invoice. Payment Terms		0..1	ASBIE	associates the invoice with the payment terms applicable/offered.
AllowanceCharge	Invoice. Allowance Charge		0..n	ASBIE	associates the invoice with an overall charge or allowance.
ExchangeRate	Invoice. Exchange Rate		0..1	ASBIE	associates the invoice with an exchange rate. In any one invoice there is only one exchange rate needed, either between invoicing at tax currency, or between pricing and invoice
TaxTotals	Invoice. Tax Totals		0..n	ASBIE	associates the invoice with summary information for a particular tax.
LegalTotals	Invoice. Legal Totals		1..1	ASBIE	associates the invoice with a set of totals required for the invoice to be a legal document.
InvoiceLine	Invoice. Invoice Line		1..n	ASBIE	an invoice has one or more invoice lines

Note that the TaxTotals ASBIE has an occurrence of 0..n.

The following is a section of the current UBL model describing the TaxTotals ABIE.



UBL Name	BIE Dictionary Entry Name	List	Occurrence	BIE Type	UBL Definition
TaxTotals	Tax Totals. Details			ABIE	information relating to the total tax for one type of tax, e.g. VAT (Value Added Tax) and all categories of that tax type.
TotalTaxAmount	Tax Totals. Total Tax_ Amount		1..1	BBIE	the amount of tax due for a single tax type, calculated from the sum of each of the tax sub totals (each subtotal for a separate category within that tax type)
TaxSubTotal	Tax Totals. Tax Sub Total		0..n	ASBIE	information relating to the tax sub total for one type of tax, e.g. VAT (Value Added Tax) and one category.
TaxSubTotal	Tax Sub Total. Details			ABIE	information relating to the tax sub total for one type of tax, e.g. VAT (Value Added Tax) and one category.
TaxAmounts	Tax Sub Total. Tax Amounts		1..1	ASBIE	associates the tax summary with totals information for the tax type on the invoice.
TaxCategory	Tax Sub Total. Tax Category		1..1	ASBIE	associates the tax summary with totals information about each tax category on the

3.2 Schemas

A Schema fragment showing the TaxTotals structure...

```
<!-- for clarity, sections of xsd:annotation not pertinent to the
discussion have been omitted in the following structures-->
<xsd:element name="TaxTotals" type="TaxTotalsType"/>
<xsd:complexType name="TaxTotalsType">
       <xsd:sequence>
              <xsd:element ref="TotalTaxAmount">
              </xsd:element>
              <xsd:element ref="TaxSubTotal" minOccurs="0"
maxOccurs="unbounded">
              </xsd:element>
       </xsd:sequence>
</xsd:complexType>
```

A Schema fragment showing the TaxTotals structure referenced from an Invoice document...

```
<xsd:element ref="cat:TaxTotals" minOccurs="0"
maxOccurs="unbounded">
       <xsd:annotation>
       <xsd:documentation>
       <ccts:Component>
       <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
       <ccts:DictionaryEntryName>Invoice. Tax
Totals</ccts:DictionaryEntryName>
       <ccts:Definition>Associates the invoice with summary
information for a particular tax.</ccts:Definition>
       <ccts:ObjectClass>Invoice</ccts:ObjectClass>
       <ccts:PropertyTerm>TaxTotals</ccts:PropertyTerm>
       <ccts:RepresentationTerm>TaxTotals</ccts:RepresentationTerm>
       <ccts:AssociatedObjectClass>TaxTotals</ccts:AssociatedObject
Class>
       </ccts:Component>
       </xsd:documentation>
       </xsd:annotation>
</xsd:element>
```

3.3 Instances

TaxTotals used in a sample instance of an Invoice document...

```
<cat:TaxTotals>
```

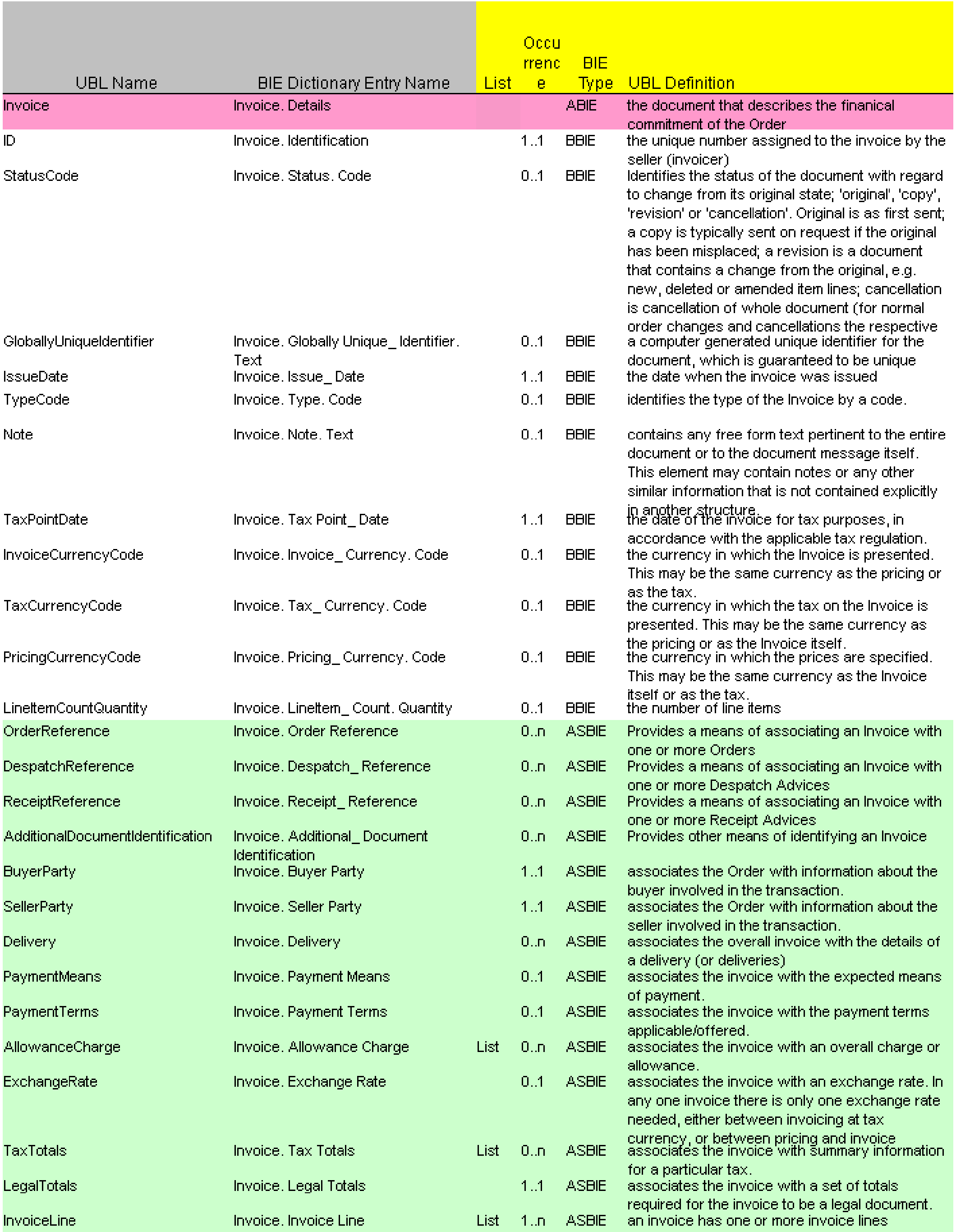
```
       <cat:TotalTaxAmount
currencyID="GBP">238.45</cat:TotalTaxAmount>
       <cat:TaxSubTotal>
              <cat:TaxAmounts>
                     <cat:TaxableAmount
currencyID="GBP">1362.56</cat:TaxableAmount>
                     <cat:TaxAmount
currencyID="GBP">238.45</cat:TaxAmount>
              </cat:TaxAmounts>
              <cat:TaxCategory>
                     <cat:ID>A</cat:ID>
                     <cat:RatePercentNumeric>17.50</cat:RatePercent
Numeric>
                     <cat:TaxScheme>
                            <cat:TypeCode>VAT</cat:TypeCode>
                     </cat:TaxScheme>
              </cat:TaxCategory>
       </cat:TaxSubTotal>
</cat:TaxTotals>

```

4 Case when using Selected List Containers

4.1 Models

The following is a revised section of the current UBL model describing the Invoice document. There are now indicators in the “List” column to denote that list containers are required for AllowanceCharge, TaxTotals and InvoiceLine.



UBL Name	BIE Dictionary Entry Name	List	Occurrence	BIE Type	UBL Definition
Invoice	Invoice. Details			ABIE	the document that describes the finanical commitment of the Order
ID	Invoice. Identification		1..1	BBIE	the unique number assigned to the invoice by the seller (invoicer)
StatusCode	Invoice. Status. Code		0..1	BBIE	Identifies the status of the document with regard to change from its original state; 'original', 'copy', 'revision' or 'cancellation'. Original is as first sent; a copy is typically sent on request if the original has been misplaced; a revision is a document that contains a change from the original, e.g. new, deleted or amended item lines; cancellation is cancellation of whole document (for normal order changes and cancellations the respective
GloballyUniqueIdentifier	Invoice. Globally Unique_ Identifier. Text		0..1	BBIE	a computer generated unique identifier for the document, which is guaranteed to be unique
IssueDate	Invoice. Issue_ Date		1..1	BBIE	the date when the invoice was issued
TypeCode	Invoice. Type. Code		0..1	BBIE	identifies the type of the Invoice by a code.
Note	Invoice. Note. Text		0..1	BBIE	contains any free form text pertinent to the entire document or to the document message itself. This element may contain notes or any other similar information that is not contained explicitly in another structure.
TaxPointDate	Invoice. Tax Point_ Date		1..1	BBIE	the date of the invoice for tax purposes, in accordance with the applicable tax regulation.
InvoiceCurrencyCode	Invoice. Invoice_ Currency. Code		0..1	BBIE	the currency in which the Invoice is presented. This may be the same currency as the pricing or as the tax.
TaxCurrencyCode	Invoice. Tax_ Currency. Code		0..1	BBIE	the currency in which the tax on the Invoice is presented. This may be the same currency as the pricing or as the Invoice itself.
PricingCurrencyCode	Invoice. Pricing_ Currency. Code		0..1	BBIE	the currency in which the prices are specified. This may be the same currency as the Invoice itself or as the tax.
LineItemCountQuantity	Invoice. LineItem_ Count. Quantity		0..1	BBIE	the number of line items
OrderReference	Invoice. Order Reference		0..n	ASBIE	Provides a means of associating an Invoice with one or more Orders
DespatchReference	Invoice. Despatch_ Reference		0..n	ASBIE	Provides a means of associating an Invoice with one or more Despatch Advices
ReceiptReference	Invoice. Receipt_ Reference		0..n	ASBIE	Provides a means of associating an Invoice with one or more Receipt Advices
AdditionalDocumentIdentification	Invoice. Additional_ Document Identification		0..n	ASBIE	Provides other means of identifying an Invoice
BuyerParty	Invoice. Buyer Party		1..1	ASBIE	associates the Order with information about the buyer involved in the transaction.
SellerParty	Invoice. Seller Party		1..1	ASBIE	associates the Order with information about the seller involved in the transaction.
Delivery	Invoice. Delivery		0..n	ASBIE	associates the overall invoice with the details of a delivery (or deliveries)
PaymentMeans	Invoice. Payment Means		0..1	ASBIE	associates the invoice with the expected means of payment.
PaymentTerms	Invoice. Payment Terms		0..1	ASBIE	associates the invoice with the payment terms applicable/offered.
AllowanceCharge	Invoice. Allowance Charge	List	0..n	ASBIE	associates the invoice with an overall charge or allowance.
ExchangeRate	Invoice. Exchange Rate		0..1	ASBIE	associates the invoice with an exchange rate. In any one invoice there is only one exchange rate needed, either between invoicing at tax currency, or between pricing and invoice
TaxTotals	Invoice. Tax Totals	List	0..n	ASBIE	associates the invoice with summary information for a particular tax.
LegalTotals	Invoice. Legal Totals		1..1	ASBIE	associates the invoice with a set of totals required for the invoice to be a legal document.
InvoiceLine	Invoice. Invoice Line	List	1..n	ASBIE	an invoice has one or more invoice lines

4.2 Schemas

A Schema fragment showing the TaxTotals structure referenced from an Invoice document...

```
<xsd:element name="TaxTotalsList" type="TaxTotalsListType"/>
<xsd:complexType name="TaxTotalsListType">
<xsd:sequence>
       <xsd:element ref="cat:TaxTotals" maxOccurs="unbounded">
              <xsd:annotation>
              <xsd:documentation>
              <ccts:Component>
              <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
              <ccts:DictionaryEntryName>Invoice. Tax
Totals</ccts:DictionaryEntryName>
              <ccts:Definition>Associates the invoice with summary
information for a particular tax.</ccts:Definition>
              <ccts:ObjectClass>Invoice</ccts:ObjectClass>
              <ccts:PropertyTerm>TaxTotals</ccts:PropertyTerm>
              <ccts:RepresentationTerm>TaxTotals</ccts:Representati
onTerm>
              <ccts:AssociatedObjectClass>TaxTotals</ccts:Associate
dObjectClass>
              </ccts:Component>
              </xsd:documentation>
              </xsd:annotation>
       </xsd:element>
</xsd:sequence>
</xsd:complexType>

```

Note that the NDR rules for global elements requires the definition of a XSD complexType for each list container. In this case, TaxTotalsListType. It is an instance of this complexType that occurs one or more times in any given document.

4.3 Instances

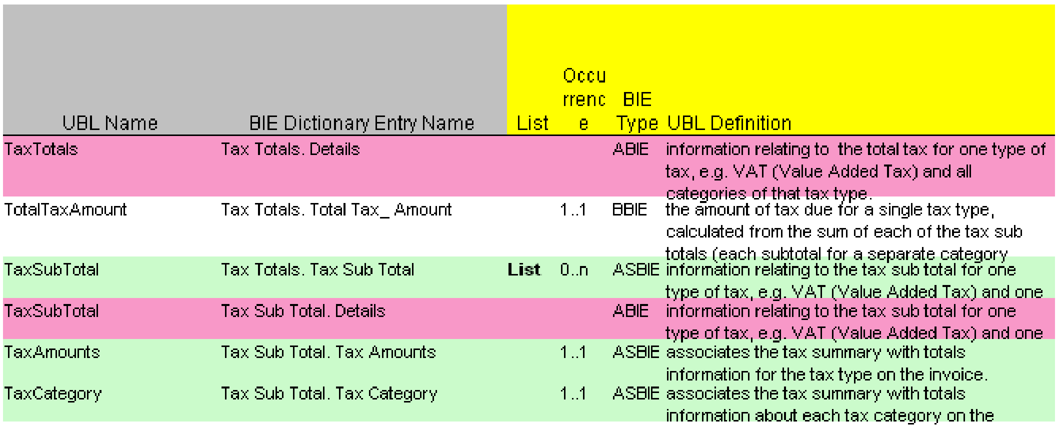
TaxTotalsList used in a sample instance of an Invoice document...

```
<in:TaxTotalsList>
       <cat:TaxTotals>
              <cat:TotalTaxAmount
currencyID="GBP">238.45</cat:TotalTaxAmount>
              <cat:TaxSubTotal>
                     <cat:TaxAmounts>
                            <cat:TaxableAmount
currencyID="GBP">1362.56</cat:TaxableAmount>
                            <cat:TaxAmount
currencyID="GBP">238.45</cat:TaxAmount>
                     </cat:TaxAmounts>
                     <cat:TaxCategory>
                            <cat:ID>A</cat:ID>
                            <cat:RatePercentNumeric>17.50</cat:Rate
PercentNumeric>
                            <cat:TaxScheme>
                                   <cat:TypeCode>VAT</cat:TypeCode>
                            </cat:TaxScheme>
                     </cat:TaxCategory>
              </cat:TaxSubTotal>
       </cat:TaxTotals>
</in:TaxTotalsList>
```

5 Case when using all List Containers

5.1 Models

The following is a revised section of the current UBL model describing the TaxTotals ABIE. There are now indicators in the “List” column to denote that list containers are required for TaxSubTotal.



UBL Name	BIE Dictionary Entry Name	List	Occurrence	BIE Type	UBL Definition
TaxTotals	Tax Totals. Details			ABIE	information relating to the total tax for one type of tax, e.g. VAT (Value Added Tax) and all categories of that tax type.
TotalTaxAmount	Tax Totals. Total Tax_ Amount		1..1	BBIE	the amount of tax due for a single tax type, calculated from the sum of each of the tax sub totals (each subtotal for a separate category
TaxSubTotal	Tax Totals. Tax Sub Total	**List**	0..n	ASBIE	information relating to the tax sub total for one type of tax, e.g. VAT (Value Added Tax) and one
TaxSubTotal	Tax Sub Total. Details			ABIE	information relating to the tax sub total for one type of tax, e.g. VAT (Value Added Tax) and one
TaxAmounts	Tax Sub Total. Tax Amounts		1..1	ASBIE	associates the tax summary with totals information for the tax type on the invoice.
TaxCategory	Tax Sub Total. Tax Category		1..1	ASBIE	associates the tax summary with totals information about each tax category on the

5.2 Schemas

A Schema fragment showing the new additional TaxSubTotalsList structure referenced from the TaxTotals ABIE...

```
<xsd:element name="TaxTotals" type="TaxTotalsType"/>
<xsd:complexType name="TaxTotalsType">
       <xsd:sequence>
       <xsd:element ref="TotalTaxAmount">
       </xsd:element>
***---><xsd:element ref="TaxSubTotalList" minOccurs="0">
              <xsd:annotation>
              <xsd:documentation>
                     <ccts:Component>
                            <ccts:CategoryCode>ASBIE</ccts:Category
Code>
                            <ccts:DictionaryEntryName>Tax Totals.
Tax Sub Total</ccts:DictionaryEntryName>
                            <ccts:Definition>Information relating
to the tax sub total for one type of tax, e.g. VAT (Value Added
Tax) and one category.</ccts:Definition>
                            <ccts:ObjectClass>TaxTotals</ccts:Objec
tClass>
                            <ccts:PropertyTerm>TaxSubTotal</ccts:Pr
opertyTerm>
                            <ccts:RepresentationTerm>TaxSubTotal</c
cts:RepresentationTerm>
                            <ccts:DataType>TaxSubTotal</ccts:DataTy
pe>
                            <ccts:AssociatedObjectClass>TaxSubTotal
</ccts:AssociatedObjectClass>
                     </ccts:Component>
              </xsd:documentation>
              </xsd:annotation>
       </xsd:element>
       </xsd:sequence>
</xsd:complexType>
```

5.3 Instances

TaxTotalsList structure...

```
<in:TaxTotalsList>
       <cat:TaxTotals>
              <cat:TotalTaxAmount
currencyID="GBP">238.45</cat:TotalTaxAmount>
              <cat:TaxSubTotalList>
                     <cat:TaxSubTotal>
                            <cat:TaxAmounts>
                                   <cat:TaxableAmount
currencyID="GBP">1362.56</cat:TaxableAmount>
                                   <cat:TaxAmount
currencyID="GBP">238.45</cat:TaxAmount>
                            </cat:TaxAmounts>
                            <cat:TaxCategory>
                                   <cat:ID>A</cat:ID>
                                   <cat:RatePercentNumeric>17.50</c
at:RatePercentNumeric>
                                   <cat:TaxScheme>
                                          <cat:TypeCode>VAT</cat:T
ypeCode>
                                   </cat:TaxScheme>
                            </cat:TaxCategory>
                     </cat:TaxSubTotal>
              </cat:TaxSubTotalList>
       </cat:TaxTotals>
</in:TaxTotalsList>
```

6 Findings

6.1 Benefits of List Containers

List Container elements foster more readable schemas and instances

Firstly, there is some doubt as to the value of human readability. Either for XSD Schema code (rather than using a schema editor) or XML instances (which are designed for applications).

Given that such a requirement exists, it is a subjective opinion whether the schemas in 4.2 and 5.2 are more readable than those of 3.2. It would depend on the environment being used.

When using an XML editing tool it is doubtful whether list containers offer any advantage as the tools tend to manage the structures independently.

When using a text editor to view 'raw' schemas it can be argued that having “lists” elements and their requisite complexType definitions create more difficulty for the reader. This is because, to satisfy the global naming rules, we flatten out the container's depth within the schema (as illustrated by definition of "ListType" global types in section 4.2) and having the schema reference those global types. Arguably, this not only offers no advantage in making schemas any easier to read than before but actually creates a more round-about reference before one finally arrives at the definition of the contained types.

With respect to readability of XML instances, it could be said that when the contained items grows into large lists, then a container could be a useful aggregation. However, with such repetitious data, humans would probably not be able to process the data visually anyway. They may then use products (such as Internet Explorer's XML viewer) that create navigable trees. In this case the list containers are useful for folding up a repeated group of elements that the reader does not want to look at.

In our study the strongest case for a list container was InvoiceLine, which repeated six times. Unfortunately, this example is too large to include in this document as it runs to over 300 lines of XML code. (This example is provided as part of the document, “UBL-Invoice-0.81-draft-8-all-list-JoineryExample.xml”).

List Container elements improve document processing performance

The opinion of the UBL TC members experienced in this area is mixed (and somewhat polarized). However, there appears to be a small favoring of list containers from 'feels right' attitude rather than any empirical evidence of benefits. This is obviously difficult to simulate in a meaningful way.

In the opinion of at least one Java/XML programmer, the main difference with processing list containers is that an application program can use the getElementsByTagName function with a list container to go more directly a group of elements. This avoids having to 'walk the tree'. Technically speaking, the results are accessed as a node (or element) with child nodes. For example, getElementsByTagName("InvoiceLineList"), provides a node with InvoiceLine children, but getElementsByTagName("InvoiceLine") produces a nodelist (or List) with Invoice Lines. The difference is between getting a node with child nodes and getting a nodelist. Usually it is the nodelist (or List in JDOM) which an application processes, ignoring any parent node. That is, the application gets a list of the Invoice Lines and just iterates through them or uses them as a collection - any container would be ignored. Consequently, it has proved difficult to find any examples where having a container parent to the children would be a benefit to a programmer.

The probable response to this point would depend on the XML processing environment being used. The ability to manipulate list containers may reduce coding logic. But there are many factors which may also impact performance. For example, validating XSD schemas with list containers would also add processing overheads as would using larger schema documents and instances. Finally, in an XSLT environment , applications would typically transform the structures to build the container structures they require anyway. The UBL Form Presentation group hold no strong opinion on the value of such structures to their processing requirements.

From a strategic point of view, we should be cautious of applying current technology issues (perceived or real) as a design criteria. Especially when, as in this case, a supplementary technology fix is available if needed. Furthermore, when we look at the design choices of other XML vocabularies (see below) it appears that not all XML designers perceive the lack of list containers as a performance overhead.

Other XML vocabularies use these constructs

A market survey reveals that the following XML business vocabularies use list containers (or something similar to):

- xCBL
- the proposed ASC-X12 XML guidelines

Whereas the following do not:

- RosettaNet
- HR-XML
- CIQ
- OAGIS

6.2 Problems with using List Containers

Naming and Re-usability

The approach taken to defining Lists is to identify occurrences of 0..n and 1..n in the model. With a few (possibly erroneous) exceptions, these appear in the identification of associations or relationships. That is, it is the Association BIEs that have occurrences of more than one. In the spreadsheet models these show as the green rows.

This is both logically and structurally correct. As an analogy, if we say a book has many pages, we do not expect the book to contain the same page many times. The book is associated with many different instances of the thing we call a page. With our Invoice example, it is not the same TaxTotals appearing several times, we expect different instances. To do this we cannot define TaxTotals as occurring many times, it is the ***association*** with Invoice that occurs many times.

Closer inspection of the schema fragments for TaxTotals in section 3.2 reveals it to be the definition of "Invoice. Tax. Totals" – not "Tax. Totals" in general. [Note the Dictionary Entry Name]

The case is clearer when we examine re-use of these list containers. Three examples sprang out of the exercise.

- TaxCategory is a potential list container (occurs 0..n) when used within AllowanceCharge and also within Item.

  However when defining the meta-data such as Dictionary Entry Name, we must use either the definition from Item.TaxCategory or AllowanceCharge. TaxCategory.

- AddressLine: the association Address to AddressLine can occur 0..7 times.

  That is, an Address may have up to seven lines. If we introduce AddressLineList containing 0..7 address lines – what happens if we want to use AddressLine more than 7 times in another association? What happens when the minOccur and maxOccur both change to 1..3 in another association? Do we define AddressLine0-7List and also AddressLine1-3List? Even though the contents within each contained items of these containers are exactly the same, there is no resuability of these container types as soon as we start defining such containers due to cardinality differences.

- BasePrice: Base price is used with three different cardinalities:

  When associated with an Item is has 0..n - “an item may have more than one base price”

  When associated with a LineItem it has 0..1 - “a item when ordered on a line may only have one price”

  When associated with an InvoiceLine it has 1..1 - “an item appearing on an Invoice line must have a price”

  If we wanted to define BasePriceList, we must do so only within its association with Item not for each time BasePrice is re-used in he library.

The findings of this reveals that the list container should be defining the ASBIE not the ABIE. Otherwise, we end up with duplicate element names for our lists.

It is not a solution to change the definitions of TaxCategory or TaxTotalsList to be generic. The BasePrice and AddressLine examples demonstrate we actually need separate list containers with their own definitions for each association (ASBIE).

Therefore, to adhere to the NDR rules, the "(name of repeating element)", should not be TaxTotalsList but should have been InvoiceTaxTotalsList. That is, we should be using the name of the ASBIE not the ABIE.

In fact, these examples in this paper had to be manually edited to avoid duplicate definitions caused by not qualifying Lists with their associations.

Identifying candidates manually (selected list containers)

If list containers may help readability in some circumstances, this then leads to the problem of how to identify what candidate structures may potentially have enough occurrences to benefit for a list container. This is demonstrated by the 'selected list containers' approach (section 4.).

With these examples, we found structures that had redundant containers (ie a container wrapping one instance), such as the TaxTotal in section 4.3. We also encountered some repeated structures that may possibly have benefited from being within a list container, but were not flagged in the models. For example, our original analysis of ItemIdentification suggested that normally one PhysicalAttribute would suffice. However, our chosen business context (the Building industry) happens to rely of these for identifying products, as can been seen from our example fragment.

```xml
<cat:SellersItemIdentification>
       <cat:ID>236WV</cat:ID>
       <cat:PhysicalAttribute>
              <cat:AttributeID>wood</cat:AttributeID>
              <cat:DescriptionID>soft</cat:DescriptionID>
       </cat:PhysicalAttribute>
       <cat:PhysicalAttribute>
              <cat:AttributeID>finish</cat:AttributeID>
              <cat:DescriptionID>primed</cat:DescriptionID>
       </cat:PhysicalAttribute>
       <cat:PhysicalAttribute>
              <cat:AttributeID>fittings</cat:AttributeID>
              <cat:DescriptionID>satin</cat:DescriptionID>
       </cat:PhysicalAttribute>
       <cat:PhysicalAttribute>
              <cat:AttributeID>glazing</cat:AttributeID>
              <cat:DescriptionID>single</cat:DescriptionID>
       </cat:PhysicalAttribute>
</cat:SellersItemIdentification>
```

Unfortunately, different industries have widely different uses of lists and entities. This demonstrates the difficulty with prediction of data occurrences and the subsequent frailty of the 'selected lists' approach.

Finally, the selective definition of list containers also creates a problem where implementors cannot immediately identify which structures might have list containers and which don't. For example, they could extrapolate whether BasePrice has a list container by looking at the models or cardinality. Worse still, they could not derive whether BasePrice had a list within Invoice and not in other occurrences.

7 Conclusion

From this exercise we have been unable to demonstrate any pronounced architectural benefits in using List Containers, either in processing performance or readability. Furthermore, even if some easily recognized value could be identified for using list containers, these would need to outweigh the difficulty of their consistent implementation into the UBL library.

We therefore suggest that Rule 116 be reconsidered by the UBL NDR team.

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.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an “AS IS” basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.