Managing Customizations of UBL

Anthony B. Coates (speaker)
Document Engineering Services

Klaus Vilstrup Pedersen
Danish National IT and Telecom Agency

Peter L. Borresen
ebConnect

Introduction

- I am Anthony B. Coates, an Associate Director with Document Engineering Services; UBL is a core speciality of DES.
- Co-authors (not present today):
  - Klaus Vilstrup Pedersen
  - Peter L. Borresen
- This presentation describes work done by DES for the Danish National IT and Telecom Agency.
Customizing UBL

- As a consultant specializing in XML standards, the majority of what I am asked to do is customization.
- Standards are a good starting point, but they often contain too much or too little (or both) for your client's business needs.
- UBL has always recognized that many users will legitimately need to customize and/or localize their UBL.
How do you customize UBL?

- If you are going to customize UBL, how do you go about it?
- What is physically involved?
- You could edit the UBL Schemas directly
- You could edit the UBL spreadsheets from which the Schemas are generated, and generate customized Schemas

http://ubl.xml.org/
How do you customize UBL?

- You could import the spreadsheets into a database, customize there, and regenerate from the database.
- That's the direct approach.
- The “quick and dirty” approach.
- In some cases, it's not the approach that you need.
How do you customize UBL?

- It's not the approach that you need if you want to re-apply your customizations to the next version of UBL, without having to reverse-engineer your changes.
- It's not the approach that you need if you want to maintain multiple customizations of UBL, and understand the relationships between those different customizations.
OIOUBL

- UBL is big business in Denmark
- The localized version, OIOUBL, is now the **only** way that you can invoice the Danish government ([http://oioubl.info/](http://oioubl.info/))
- That means a > $100 million / year saving for this country of 5½ million people
- These savings depend on OIOUBL, and so OIOUBL needs to be managed as a valuable public resource
Initially, the UBL spreadsheets were imported into a relational database, and the customized OIOUBL components were stored in the database in the same format.

However, the current format models UBL and OIOUBL as two separate component libraries.

Doesn't capture how OIOUBL is derived from UBL.
DES is developing a UML metamodel of UBL and its customizations and localizations for NITA (Danish National IT and Telecom Agency)

Metamodels are common in this kind of work – a model of what is allowed and supported in your model
However, metamodels often support only snapshot views of models – as that is the simplest place to start.

It is more complicated to produce a metamodel that deals with versioning of all components, etc.

It is more complicated to produce a metamodel that supports modelling of customizations in a re-usable format.
Modelling Customizations

- How do you model customizations?
- The new metamodel for OIOUBL (and other derivatives of CCTS) breaks customizations down into the following categories:
Modelling Customizations

XML-in-Practice 2008
Modelling Customizations

- Simple Customizations:
  - Annotation – addition of documentation or metadata to a library component, without changing the underlying component
  - Rename – the component is renamed, but not changed otherwise
  - Extension – extra sub-components are added to the component
  - Replacement – the component is completely replaced
Modelling Customizations

- Simple Customizations:
  - Reduction – a component is derived by removing something from the original component
    - Restriction – a narrower component; all instances of this component are also valid instances of the original component
    - Reduction – some sub-components removed completely (a “projection”)
Modelling Customizations

- Extensions are supported by many software tools, particularly object-oriented languages like Java, C#, C++, etc.
- Also supported by W3C XML Schema
- New sub-components added/appended to existing component
Modelling Customizations

- By contrast, restrictions are not well supported – but are important to CCTS-based models.
- In a restriction, you can decrease the upper-bound multiplicity or increase the lower-bound multiplicity, or narrow the type.
- Repeated $\rightarrow$ singular, optional $\rightarrow$ mandatory.
- W3C XML Schema supports this.
Modelling Customizations

- Reductions are different again – in a reduction, you remove sub-components completely if you don't want/need them:
Modelling Customizations

Standard Message  
[Global and Automotive]

Field #1 [Global]  
Field #2 [US]  
Field #3 [EU]  
Field #4 [Italy]  
Field #5 [Transport]  
Field #6 [Airlines]  
Field #7 [Shipping]  
Field #8 [Trucking]

Customised Message  
[EU and Airlines]

Field #1 [Global]  
-- removed --  
Field #3 [EU]  
Field #4 [Italy]  
Field #5 [Transport]  
Field #6 [Airlines]  
-- removed --  
-- removed --
Modelling Customizations

- It is important to support reduction when your model has to support XML messages as well as databases and application models.
- A message usually only needs a subset of the total content.
- It's not a restriction, it's an ad-hoc subset that is appropriate for a particular task or context.
Modelling Customizations

- These are the various simple customizations that need to be modelled.
- In addition, a “complex customization” is an application of multiple customizations (simple or complex).
- When you model customizations this way:
  - You can re-apply them when necessary.
  - You have an audit trail of what has changed, and how.
Non-destructive Customization

- Here's an analogue
- Many photo-editing apps now support non-destructive editing of photos, e.g.
  - Adobe Photoshop Lightroom
  - Google Picasa
- These packages store a set of changes to each photo, not a changed photo
- The changes can be undone, modified, and/or applied to other photos
Non-destructive Customization

What has been described in this presentation is something similar

Non-destructive customization of component models
  - You don't lose your original component
  - You can undo or modify the changes
  - You can reapply the changes or apply them elsewhere
Non-destructive Customization

- There is now a plan to open-source the metamodel and create a set of open-source software tools to support non-destructive editing of component models.
- “Open Information Management”
- If this interests you, please talk to me afterwards, or contact one of us later by e-mail.
Contact Details

Contact details:
- Anthony B. Coates - anthony.coates@documentengineeringservices.com
- Klaus Vilstrup Pedersen - kvp@itst.dk
- Peter L. Borresen - plb@ebConnect.dk