Thoughts about UBL library customization

Jon Bosak
Chair, OASIS UBL TC

Hong Kong
11 & 13 May 2004

http://oasis-open.org/committees/ubl

Sun microsystems

OASIS

About these slides

The first part of this slide set shows the opinions I delivered at the beginning of the Tuesday plenary 11 May 2004 in Hong Kong. The second part gives my preliminary conclusions based on the Tuesday discussion.

In the interim, the members gathered in the tech track have discussed this further. I was unable to participate in that discussion.

This slide set represents my input to the continuation of the library customization issue scheduled for Thursday afternoon in Hong Kong. It is a temporary document not intended for use outside the context of that discussion.

See the report of the Friday plenary for the outcome of this discussion.

It's about economics

- EDI works
- But it's too expensive
- Because every installation is a custom installation
- Most experts believe that businesses will continue to insist on extensive customization
- But that's because businesses have not been given *cheap off-the-shelf software* as an option

No crystal ball needed

I've already seen this movie several times:

- When optimized system software stopped being produced by custom software houses (1960s)
- When expensive quality typesetting software was replaced by crappy desktop publishing systems (1980s)
- When powerful, flexible hypertext publishing systems were replaced by extremely limited free web browsers (1990s)

The lesson of commoditization

Given the choice between

- Powerful, flexible, high-quality, expensive software that meets all their requirements and
- Cheap, inflexible, limited, commoditized off-the-shelf software that meets just their most basic requirements

people will choose cheap commodity software *every time*

and then figure out ways to work around the worst of its limitations and live with the rest

A new choice for B2B

You haven't seen this happen in B2B yet because people haven't been given a similar economic choice.

UBL 1.0 is going to give them that choice.

You may say "businesses will always have a set of customization requirements."

Let's see what happens to those requirements for custom solutions when the commodity alternative costs 1/10 or 1/100 or 1/1000 as much.

Let's see what happens to those requirements for custom solutions when governments begin to dictate standard input formats to 10,000 suppliers at a time.

A standard markup ecosystem

- Cheap commodity software (example: web browsers)
- Standardized training (example: HTML)
- The magic conversion of difficult generic XML tools requiring expensive consultants into easy-to-use tools driven by stock configuration files
  - XSL-FO output formatters driven by off-the-shelf stylesheets
  - XForms input tools driven by off-the-shelf XForms
  - Adobe eForms templates to do the same thing
- The emergence of less flexible applications hardwired to behave the same way (example: OpenOffice plug-ins)
- Bazillions of little ad hoc VB programs, Java programs, perl scripts, shell scripts, word processing macros, hacks, freeware, shareware...
- Bazillions of stock examples, lessons, assignments, programming cookbooks, toolkits, FAQs, user forum tips and tricks...
- *A developer and user community that speaks that language and develops a religious commitment to that language*

The meaning of conformance

Instance conformance in a commodity ecosystem means exactly one thing: *the instance doesn't break any of that stuff.*

- All the free software keeps working
- All the free stylesheets and input forms keep working
- All the plug-ins keep working
- All the little ad hoc scripts and tools keep working
- All the examples still apply
- All the little hacks and tricks stay usable
- All the lessons and assignments and tests are still valid
- All the people who can claim to be experts maintain their reputations
- In sum, there are no surprises.

The point of XML validation is to guarantee that I won't be surprised.

And don’t forget conversion

We’ve heard a lot over the last day about efforts to develop conversions to and from existing data formats. We can’t break all that conversion software, either.

The fork in the road

In a standard markup ecosystem, the critical difference will be between customizations achieved through *restriction* to the UBL 1.0 schemas and customizations achieved *any other way.*

In software terms, the thing to be feared is uncontrolled feature creep.

Our problem is how to accommodate the *genuinely unavoidable* requirements for extension without breaking anything.

Example: The addition of “InspectionDate” to Invoice (a requirement of Japanese commercial law).

An obvious beginning

- Do everything possible by restriction
  - Look at our experience so far from ECOM!
- Which means: incorporate the really necessary additions directly into the standard via periodic new releases
- Then the problem becomes: how to add stuff in a controlled way without making the standard set difficult to implement

Some initial thoughts

- Incorporate small-scale changes into point releases (definition of point release: all the old instances continue to validate)
- Handle large-scale or structural changes by defining new document types (example: Australian grain status despatch advice) and give them experimental or provisional status until ready to include them in some future release
- In the 1.0 to 1.1 time frame:
  - Focus on deployment of 1.0 (including L10N)
  - Gather and consider use cases one at a time with a view toward distinguishing “nice to have” from “need to have”
  - Make understanding the problem our primary goal
  - Let those who customize beyond restriction do what they like and learn from their experiences, but don’t let them call what they produce “UBL 1.0”
  - Consider formally proposed additions (e.g. COO) on a case-by-case basis and be ready to approve them for 1.1 if it clearly makes sense to do so

Discussion

After I presented the foregoing slides on Tuesday, we discussed the use cases originally developed by the TTSC (see separate document) and Tim's later summary use case descriptions. In the following slides, I present my own conception of each use case as it would apply to our work in 2004 as my input to the Thursday afternoon plenary.

Use case 1

- The UBL 1.0 schemas are used as provided in the CD.
- Action for UBL TC: none.

Use case 2

- An XYZ industry profile is developed by defining XYZ schemas that are proper subsets of the UBL 1.0 schemas. The definition of “proper subset” is that any valid XYZ instance is also a valid UBL 1.0 instance.
- Action for UBL TC: Because the XYZ instances will carry a non-UBL namespace, we need to (or should) develop a simple technique whereby XYZ instances can be made to look to off-the-shelf UBL 1.0 applications like UBL 1.0 instances. Perhaps this could take the form of a configuration file recommended for inclusion in every conformant UBL 1.0 processor that will allow it to recognize that the XYZ namespace is in fact a subset of the UBL 1.0 namespace and substitute the UBL 1.0 namespace for the XYZ namespace as the first step in instance processing. (We didn’t spend more that about 30 seconds on this; it’s logged here simply so that we don’t forget it.)
- Question to be resolved: Can the XYZ schemas be called “UBL”? (My tentative opinion: yes, given a standard solution to the namespace problem just referred to. See the CM Guidelines for more on this.)

Use case 3

- The XYZ industry submits new document types to be considered for inclusion in the next UBL release. Example: The Certificate of Origin documents submitted by Crimson Logic.
- Action for UBL TC: develop a procedure by which such submissions can be processed case by case based on the decision tree traversed by XYZ in forming the new document types. Such a process should tell us the following (for example):
  - Which library schema modules needed to be changed?
  - Were those changes simple additions or were they structural modificatons?
  - If simple additions, should we perhaps make those additions to the existing library component, or should we define the variant as a new library component?
  - If structural, do we accept the new variant into the library?
  - And so on
- In my view, this is primarily a committee process problem and not a design methodology problem.

Use case 4

- The XYZ industry wishes to build their own schemas using whatever they like out of UBL but not with the intention of submitting their work for inclusion in a future version of UBL. There are three variations of this case.

  4a. XYZ follows the UBL NDRs and uses the library modules unchanged. We agree that this case is highly unlikely.

  4b. XYZ follows the UBL NDRs. In this case, they can say "the XYZ schemas were developed in accordance with the UBL NDRs." *As we in UBL are agreed that we do not intend to provide UBL certification,* no action on our part appears to be required. [Mark: add: however, we have said that we would provide a methodology for this and that they could then claim they were "UBL conformant." So this is an action for us.]

  4c. XYZ does not follow the UBL NDRs. In this case, some may feel it desirable to provide guidelines to ensure or promote or foster some kind of "UBL alignment", but personally I doubt that we will be able to formally define this concept, and in any case I don't see this as something upon which we should be expending resources as we work toward 1.1.

My conclusions from Tuesday

- We need to figure out a strategy for dealing with different namespaces in instances generated by applications working with a subset schema (industry profile).
- We need to develop a procedure for considering submissions to future versions of UBL that includes a decision tree for submitting organizations that will provide us with a clear picture of the changes they made in creating their new document types.
- It may or may not be desireable to create “customization guidelines” for organizations that don’t intend to submit their creations to UBL, but in any event
  - We don’t have the resources to attack this in 2004
  - And we don’t understand the problem well enough to do a good job if we did have the resources
- My recommendation for 2004: focus resources on UBL 1.0 deployment, concentrate on solving the first two problems above, and learn from what organizations do with UBL 1.0.

An immediate problem: interim extensions

- At this meeting we've discovered that we have a genuine "customization" problem right in front of us: the fact that Japanese commercial law requires Inspection Date in invoices. What do we do about this?
- Initial thought for how to deal with such cases (there will be others):
  - Discuss and formally agree on the "accepted addition" to UBL 1.0 and add it to a public list we maintain of items we know will be included in UBL 1.1
  - Publish this decision to the ubl and ubl-dev lists as an advisory to implementors to include the capability to handle this addition in any UBL software that is scheduled for eventual conformance to UBL 1.1
  - Assign a namespace to be used on (say) UBL invoices in Japan that indicates that invoice instances so labeled may include additions made up to the date of approval embedded in the namespace
- This is just an initial thought. My point is that **it represents the kind of problem we should be trying to solve right now.**