Do It Yourself (DIY) XBRL - Is There a Better Approach

I've been working with global XBRL implementations over the past eight years and sometimes get into discussions with experienced software developers in regards to the "buy or build" argument. Obviously this argument isn't unique to XBRL and, being a software developer myself, I totally understand developers questioning the need to use a specialised XBRL package to create "a simple XML document".

Typically I hear comments which can be summarised as - "All we need to do is build a simple XML handler which can generate the XBRL instance documents which SuperStream mandates. They're just XML documents which need to conform to a schema. How difficult can that be?"

While understanding such comments, I don't believe that the requirement to create XBRL instance documents consistently and reliably is quite that simple. This blog is my response to the questions which have been raised with me over the last eight years and, I am happy to admit, reflects my personal opinion. I would also like to highlight that I have more than a personal interest in the topic as the company I work for has a set of commercial XBRL offerings!

The comments below reference the Australian SuperStream taxonomies, however they are also relevant for any XBRL reporting regime.

Summary

What is required is more than just the ability to generate simple XML output. The DIY processor needs to properly understand the requirements of XBRL (e.g. dimensions) to reliably create and consume XBRL instance documents and this requires a great deal of effort, more than you might expect. The processor also needs to be scalable to handle likely increases in the size and number of XBRL instances.

Any document created must be checked for XBRL compliance so that it won't be rejected when it reaches its destination. Schema validation is definitely not enough, validation requires full XBRL validation. Consider the business impact and delays if documents are rejected at the destination.

Taxonomies are dynamic and so the impact of changes to taxonomies needs to be minimised. A properly designed application with data mapping between source data and XBRL data will greatly assist maintainability.

The XBRL standard is evolving and so is the use of enhancements within reporting regimes. How would your DIY solution cope with the need to incorporate XBRL Formula support?

Effort Required in Building and Maintaining a Reliable XBRL Processor

While creating really simple XBRL instances with Java/C#/C++ and your favourite XML parser, the reality is that most XBRL documents are simply not that straightforward. (If they were then there would be no need to use XBRL!)

As an example, consider the way XBRL items ("facts") are related, via XBRL contexts, to dimensions. A very common error which occurs is the notorious "[Ins Err, 2] The primary item contains invalid hypercubes in all base sets" error. This can occur for a number of reasons and can be very difficult to trace. But, if a DIY XBRL processor does not fully support the XBRL Dimension 1.0 specification, then it is very unlikely that this error will be detected during the construction of the XBRL instance. Rather the error will be detected when the instance document is received by a third party who will then reject the document. In contrast, a commercial XBRL processor will automatically detect the error during instance document creation and highlight the problem.

Scalability and performance are other issues that need to be considered. When documents are small performance is not usually an issue. But experience has shown that, once a document reporting regime (e.g. XBRL) is implemented, documents start to increase significantly in size. In SuperStream the member rollover allows for multiple rollovers per instance. MCS instances are likely to be even larger. Any XBRL implementation therefore needs to be able to deal with potentially large documents. Commercial packages have inbuilt mechanisms to reduce the memory footprint as they internally build an XBRL model as opposed to the DOM model which has greater memory requirements. While SAX can be used to lower the memory requirement it needs to be remembered that the order of items and the contexts they refer to cannot be guaranteed within an XBRL instance.

Validating the XBRL Instance Document

Okay, so you've written your XBRL instance document using your favourite DOM and SAX parser and it looks valid to you. But can you be sure that it is fully XBRL compliant? Even if your application just changes the value of an element (rather than add new elements and values) can you be sure that your DIY solution will produce a result that is valid XBRL?

Take the case of explicit dimensions. The XBRL dimension specification requires that, for an explicitDimension element, the dimension attribute contains the name of the dimension and the domain is the value of the element itself. Both of these need to match a valid combination of dimensions and domains defined in the taxonomy. If an invalid combination (or even worse a non-existent value) is accidentally entered then the document is not XBRL compliant for the specified taxonomy. But this cannot be validated simply by using schema validation as the dimension requirement is contained within the definition linkbase, not the schema.

However, a compliant XBRL processor will validate all of the dimensional requirements and would ensure that such an error was reported before the document was lodged.

The SuperStream taxonomies use the SBR taxonomy architecture for XBRL. This architecture includes a number of rules which effectively extend the XBRL rules. For example, they stipulate that all items within a tuple must belong to a single context (this includes child tuples and their items). So it would be advantageous to use an XBRL processor which not only conforms to XBRL rules but also to the SBR rules. But it is important that such an extended processor be based on a robust XBRL processor as the SBR and XBRL rules are complementary.

A simple example might illustrate the issue best. In December 2011 year the initial exposure drafts for the SuperStream Member Registration and Member Contributions were released. Because of a complex issue relating to primary item/hypercube inheritance it was not possible to create a valid XBRL instance document using these taxonomies. Given the complexity of that issue, I believe that it is very unlikely that a DIY XBRL solution would notice and highlight the problem. Rather it would probably generate what it believed to be a valid XBRL instance document - which is actually not possible. However, this issue was easily detected by the XWand XBRL processor used in the SBR API. (FYI, the SuperStream October 2012 exposure drafts have addressed this issue.)

A final thought on validation. If you don't have an XBRL processor which can fully validate an XBRL instance according to both the XBRL 2.1 and Dimension 1.0 then, as noted above, you can't be sure that your document is XBRL valid. If that document is a SuperStream Member Rollover you'll only have 3 days to finalise the member transfer. Let's assume that there is a minor XBRL error in an instance document. You send it to your gateway which passes it to another gateway and finally it arrives at the destination. Shock! horror! They reject the document and send an error response back through the gateways to your system. Do you now have time to analyse the error, work out what is wrong and deliver an updated instance (hopefully valid this time!) before the mandatory three days has elapsed?

Changes to the taxonomy

Compliance and business requirements are subject to ongoing change and so taxonomies need to be updated to match those changes. Whether this happens annually, quarterly or even more often, the effect is still the same - the software which creates the XBRL instance documents needs to be modified. If that software uses a commercial XBRL package then the impact can be minimised by adopting a data mapping process between an organisation's data and its XBRL representation. If, however, the process of creating the XBRL documents is embedded within the application then the process of updating can easily become more complex and costly.

Work is currently being carried out within the XBRL consortium on a Versioning specification. Once this is complete it will enable users of XBRL packages who implement this specification (most commercial packages are expected to) to easily understand, in a programmatic way, the differences between two taxonomies. This will enable smarter applications which can detect and allow for differences between versions of taxonomies.

Changes to the XBRL standard

Like taxonomies, the XBRL specification is constantly changing. In particular, new specifications are released and, inevitably, taxonomy designers take advantage of these new specifications.

A good example of this is the XBRL formula specification which was ratified in 2009. Currently, the SuperStream taxonomy does not use XBRL formulae but it is entirely possible that this could change in the future. Would your DIY XBRL processor easily handle XBRL formulae which are expressed as a series of function calls and XPath expressions embedded in a completely new set of roles and arcroles? If it can't do this then how are you going to validate your instance document against the rules expressed in the XBRL formulae before submitting it?

Conclusion

I believe that you have to consider whether your core expertise should include XBRL or should be focussed on your application domain - superannuation, accounting etc. Then you should carefully analyse the business case for buying XBRL software versus the cost (and risks associated with) developing and maintaining your own XBRL solution.

If you'd like to discuss this article or question any of my thoughts, comments, etc. please send an email to peterc@fast.fujitsu.com.au

I look forward to hearing from you!

Peter Campbell

Principal Architect

Fujitsu Australia Software Technology