Comparing XML and XBRL

By

Cliff Binstock, UBmatrix, USA (Cliff.Binstock@UBmatrix.com)

Charles Hoffman, CPA, UBmatrix, USA (CharlesHoffman@olywa.net)

Rene van Egmond, UBmatrix, USA (Rene@c3ismc.com)

Phil Walenga, UBmatrix, USA (Phil.Walenga@UBmatrix.com)

July 9, 2005

Purpose and Audience

One of the first questions many people ask about XBRL is how it is different than XML. While XBRL is XML, there are differences. XBRL provides features above and beyond what XML provides. This white paper tries to answer the question, "What are the differences between XBRL and XML?"

This white paper is intended to communicate to both technical people and business people who have at least a little technical understanding of concepts like XML, semantics, normalization, etc.

* * *

Table of Contents

Summary 2
Details 3
XML 3
Prescriptive Extensibility.. 4
Express Semantic Meaning 5
Content Validation of Semantic Meaning 5
Normalized.. 6
Express Multiple Relations 6
Fits into Relational Databases 7

* * *

Comparing XML and XBRL

Many people ask the question, "What is the difference between XBRL and XML?" First, XBRL is XML; but XBRL adds to XML and uses XML in a somewhat unique way to meet the business requirements it must fulfil.

Summary

The following is a summary of the features of XBRL. Each of these features is further discussed and elaborated on in the details section:

- **XML**: XBRL is XML.
- **Prescriptive Extensibility**: XBRL's powerful flexibility is achieved through prescriptive extensibility, rather than using the both overly constraining and overly flexible extensibility methods of native XML. It still allows XML's flexibility which can be leveraged in your applications; it is just that an XBRL processor will not use those pieces as it won't understand what to do with them.
- **Express Semantic Meaning**: XBRL provides a robust method of expressing semantic meaning (for example, agreed upon business relationships), such as "Assets = Liabilities + Equity". This feature does not exist in native XML and probably never will.
- **Content Validation of Semantic Meaning**: XBRL provides instance document content validation of semantic meaning, or business rules, using the expressed semantic meaning (again, something not achievable with native XML because there is no method of expressing semantic meaning; you would have to build it yourself).
- **Normalized**: XBRL is a normalized version of XML (that is, the lack of normalization in native XML makes it difficult to extend in a consistent way).
- **Express Multiple Relations**: Because XBRL is normalized, XBRL has the ability to express multiple explicit relations for use with semantic validation, formatting, etc. as opposed to being limited by the one implicit relation provided by native XML's content model. Try and express semantic meaning

within the one content model XML provides, you quickly realize the limitation of native XML.

- **Fits into Relational Databases**: XBRL was built to provide an ability to work well with a relational database, because it is normalized and because it is rather "flat". Lots of people use relational databases.

Several other points are worth mentioning about XBRL. First of all, XBRL is not really intended to be used without an XBRL processor. XBRL processors understand all these additional features provided by XBRL. XML parsers don't. While you could do it, it is not that practical. One would have to dumb-down what you are trying to achieve so much that you would miss many of the useful features of XBRL. Or, you would basically have to build the functionality which is already in an XBRL processor to use the features; therefore you literally end up creating your own XBRL processor.

Open global standards are rather expensive and time consuming to create, especially highly functional languages, such as XBRL. Native XML cannot solve the needs of business reporting, to do so one would have to create an XBRL like XML language to do that. To achieve what XBRL has achieved, it would literally have to be reinvented which took years to agree on, create, let alone implement. But what would you have when you are done? It probably would be something like XBRL, perhaps with somewhat different syntax.

Lastly, one of the values of XBRL is standardized taxonomies; particularly taxonomies like the IFRS-GP taxonomy and US-GAAP taxonomy framework, both of which express semantics for financial reporting. It took thousands of hours to create these two taxonomies, probably more man hours than creating the XBRL specification itself. The point is that this value is above and beyond having XML to create the semantics in.

XBRL brings the software closer to the end user. Software which works with all XML software is typically developer tools, such as XML Spy. Off the shelf software is beginning to be created to use XBRL. Lots of this software will be end user software which will work with any XBRL taxonomy or instance document.

Details

Now we elaborate on the statements made above, explaining the statements in order to allow the reader to understand why the statement is true. We also explain the business case behind each statement.

XML

XBRL is XML.

XBRL uses XML, XML Schema, XLink, XPath, and other W3C XML standards heavily. XBRL is very committed to XML. But, native XML (an XML instance validated by an XML Schema) is only a partial solution to what XBRL needs to achieve. Also, some of XML's powerful features – its strengths – in one circumstance are also its greatest weaknesses given other circumstances.

A very powerful feature of XML is validation of a content model to ensure it is correctly expressed: the elements and attributes are properly arranged and you are not missing an angle bracket; that the document is syntactically valid. This is achieved using XML and XML Schema. XBRL leverages these powerful features. Data type validation is also achieved via XML Schema, for example a specific element has to be a number, not a string. This is validation of the syntax of the XML. XBRL does give up part of the content model of XML, XBRL is rather "flat". The reason for this will be discussed later.

One thing about XML which is very powerful is its ability to express a rich content model. But at the same time, that content model expresses only implicit relationships, rather than explicit ones. With XML you can express relationships between concepts in your document, but you cannot express how they are related, or if two different types of relationships exist, the fact that they are different cannot be expressed. Also, you can only express one relationship: the content model.[1]

Another powerful feature of XML is its extensibility. Again, this extensibility becomes a problem if the extensibility is used by the producer of information, and the consumer of that information does not expect those extensions.

Since accuracy of the data is so important and XML only provides syntactic validation of the data, XML can only be a partial solution to improve business reporting.

From a business perspective, the point is this: XML can only be a partial solution while XBRL can provide a complete solution to fully automate business reporting. If XML alone is used, each user of XML will have to add additional functionality to any XML solution already provided by XBRL. To meet your proprietary needs, a user of XBRL still may need to add functionality not offered by XBRL, but that list of functionality which would have to added would be longer had XML alone been used.

Prescriptive Extensibility

XBRL's powerful flexibility is achieved through prescriptive extensibility, rather than using the both overly constraining and overly flexible extensibility methods of native XML. It still allows XML's flexibility which can be leveraged in your applications; it is just that an XBRL processor will not use those pieces as it won't understand what to do with them.

One of the most powerful features of XBRL is a robust prescriptive extensibility mechanism, yet flexible in the areas XBRL needs to be flexible. Using this mechanism, users of XBRL can add financial concepts and/or add or remove relations from an XBRL taxonomy. Two things are achieved. First, producers of information can take an existing taxonomy and modify it for their specific reporting needs and second, the changes made by the creator of the information are clearly documented in their extension taxonomy.

An XBRL processor is required to understand extensions, and extensions using XBRL standards will not break the software.

Additionally, the flexible XML extension mechanisms can still be used, but they will likely be used for proprietary purposes, where the producer knows who will be consuming the information and the consumer's software can take advantage of the new elements or attributes added to XBRL.

From a business perspective, the point is this: XBRL provides flexibility where you need it, whereas XML provides too much flexibility where you don't. XBRL allows an end user to take a standard XBRL taxonomy like US GAAP Insurance and make changes to it, extend it, or use a subset of an XBRLized regulatory report like Basel II for bank capital reporting and still use standard software tools. This would not be

[1] XML Schema provides *identity constraints* which provide limited semantic support that mimics "relational integrity" in a database.

possible with an in-house developed XML language for business reporting. In addition, you can still add the limitless flexibility offered by XML to XBRL for your internal proprietary needs. XBRL can be both flexible and not break existing applications; it was architected to do just that.

Express Semantic Meaning

XBRL provides a robust method of expressing semantic meaning (for example, agreed upon business relationships), such as "Assets = Liabilities + Equity". This feature does not exist in native XML and probably never will.

Another very powerful and robust feature of XBRL is the ability to model semantic meaning, which can be thought of as a way of saying that one can model business rules, for example. This is a very important requirement desired and even demanded by users of XBRL. For example, the Federal Deposit Insurance Corporation (FDIC), which is an early adopter of XBRL has 1800 business rules which needed to be run against data being captured. As they say, "Garbage in, garbage out." XBRL users exchange a great deal of numeric information they require to be accurate and complete.

The semantics which need to be expressed can take different forms. Calculations to handle summations are one form, a formula is another, and the definition linkbase can contain others. The content model of XML allows for the creation of only one content model, and that implies relations rather than making them specific. The important thing is that the content model actually gets in the way of expressing semantic meaning in XML.

From a business perspective, the point is this: The data used in business reporting has a lot of semantic meaning. With XBRL you get a way to express this meaning "out of the box", so each user does not have to create them. Because each user, for example, can reuse the 1800 XBRL business rules to validate bank regulatory reports it is less expensive for everyone. Also, users don't have to reinvent a scheme for expressing the semantic meaning; XBRL provides that scheme which is used by all XBRL software.

Content Validation of Semantic Meaning

XBRL provides instance document content validation of semantic meaning, or business rules, using the expressed semantic meaning (again, something not achievable with native XML because there is no method of expressing semantic meaning; you would have to build it yourself).

Because the semantic meaning can be expressed, content validation against this semantic meaning is possible. XBRL provides this also. The example above, "Assets = Liabilities + Equity" is impossible to validate using XML or XML Schema. XML parsers might validate that "Assets" is numeric, or that it must exist, but XML cannot validate the complex semantic meaning which exists in business reporting. XML does not even have a way of expressing the meaning; users have to individually build these solutions above and beyond what you get from simply implementing a solution using just an XML Schema.

From a business perspective, the point is this: Because XBRL has expressed the semantic meaning (see the above section), users can use this semantic meaning to test the data. Quality of the data is improved. What you get with XBRL is a "rules engine" based solution which is a "one-to-many" solution. Many users can reuse the one solution. XML semantic validation can only be built "one-to-one"; basically each implementation needs to build its own. This is part of the reason the FDIC projects it will save $27 million using XBRL to capture data from US banks.

Normalized

XBRL is a normalized version of XML (meaning that XML is not normalized, which causes extensibility issues).

XBRL is normalized, similar to what it means to normalize a relational database, a process makes processing more efficient. What this means is that there is a separation between XBRL concepts (which are defined in Taxonomy—XML Schema—files) and relationships (which are defined in Linkbase—XML—files). Trying to put calculation information, presentation information, definition information, label information, reference information into one (or one set) of XML Schema files would be challenging at best. Trying to then extend this information once it has been expressed would be even more challenging, for example, "Assets = Liabilities + *Commitments* + Equity".

However, because XBRL is normalized, it is actually very straight forward when it comes to trying to rewire a calculation, move or prohibit a presentation, etc.

From a business perspective, the point is this: Because XBRL is normalized, somewhat like the data in a relational database is normalized (not exactly, but you can think of it in these terms), XBRL is more flexible. While XBRL is often considered too complex, business users understand that business reports can be complex and expressing them in a standardized can be technically complex. Ignoring the business requirement of extensibility is not an option for XBRL. However, tools have been created to reduce the complexity to the end user, and easier and more powerful tools will be coming. Normalization leads to other benefits and features that offer advantages compared to XML as noted below.

Express Multiple Relations

Because XBRL is normalized, XBRL has the ability to express multiple explicit relations for use with semantic validation, formatting, etc. as opposed to being limited by the one implicit relation provided by native XML's content model. Try and express semantic meaning within the one content model XML provides, you quickly realize the limitation of native XML.

This is the result of normalizing XBRL. Literally an infinite number of different relations or resources can be expressed in XBRL. The key core relations are provided: presentation, calculation, definition. Others can be easily created using the same mechanisms which XBRL uses. This architecture will really make it easy for XBRL to evolve.

With native XML, again, you get one hierarchy, the content model of XML.

While "multiple relations" is not really the totally correct term, it is used because most people understand the term relation or hierarchy or "tree". More precisely, XBRL allows a user to model an "n-dimensional" (infinite number of dimensions) graph, rather than just a tree. Graphs can contain cycles, tree hierarchies cannot. XBRL information should be viewed more like a "web" than a "tree".

From a business perspective, the point is this: Because of XBRL's flexibility it can handle multiple relationships and better express complex business reports. Trying to do this with XML would be problematic. In addition, most users of XBRL will have proprietary needs in addition to the base which XBRL offers. Fulfilling these proprietary needs is easier because literally an infinite number of different types of relations or resources can be expressed.

Fits into Relational Databases

XBRL was built to provide an ability to work well with a relational database, because it is normalized and because it is rather "flat". Lots of people use relational databases.

XBRL provides an infrastructure for modeling relational data. Lots and lots of people have relational databases which contain business information which will be used to generate XBRL instance documents and/or to capture business information and store it after it has been received from someone else.

Because XBRL is normalized, it is quite easy to import information from XBRL taxonomies and instance documents into relational database tables.

From a business perspective, the point is this: It is important for XBRL to work well with relational databases because (a) there are a lot of relational databases in existence, (b) that is where much of the data which is exchanged is stored, (c) that is where data received from others will likely be stored. Exchanging financial data between relational databases and XBRL is efficient, reducing costs of automating the process.