Why Is XBRL So Hard?

by David vun Kannon

I'D LIKE TO SHARE MY THOUGHTS ON WHAT makes XBRL (eXtensible Business Reporting Language) difficult to understand when people approach it for the first time. This is my perspective, which is based on my participation in the design of XBRL from its beginnings and the many attempts to explain XBRL to different audiences.

Why should you care how hard XBRL is to understand? It's a technology that often has to be understood (at least partially) by nontechnical people who have to judge the impact it might have on their business processes. Depending on the industry, those processes may be more or less fundamental to the business. The impact of XBRL can vary from insignificant to small but crucial, radical, and transformative. Individuals with more technical responsibilities need to understand it at an even deeper level.

To me, the following are the leading conceptual difficulties about XBRL today. I've also included a coping strategy for each concept. Some of the strategies relate to how to think about your business problems while others are more specific to software developers.

SCOPE ● The scope of the business problem that XBRL is trying to address is very large. Most developers and users have smaller problems or more constrained problems, so some of the features of XBRL look like overkill from that perspective. *Strategy:* Use the features you need, and leave the rest. Remember, someone else needs the feature you don't! Think about how your application could fit into a larger environment.

DESIGN ● XBRL is two different languages. It's a language for communicating business facts (XBRL instances) *and* a language for communicating the "metadata" context of the business fact (XBRL taxonomies). While the languages interoperate, they have very different designs. *Strategy:* Most users will deal with the simpler language of facts far more often. But metadata management is a key problem in consolidation and corporate reporting, so it's important to understand how that part of XBRL can help document processes.

EXTENSIBILITY ● Both XBRL languages have strong extensibility. In the fact language, the business segment, reporting scenario, and even unit of measure are basically wide open. In taxonomies, new linking roles and arc roles can create entirely new intellectual structures. *Strategy:* You don't have to understand all of the basic roles the first time around, and the rules of XBRL say that you can ignore extension materials without fear of losing anything fundamental. XBRL really is modular, though its modularity hasn't been explained well.

EDGE ● While the fact language is a fairly straightforward XML language, the taxonomy language is much more "bleeding edge." Taxonomy development software must deal with inconsistent implementations of core technology by vendors. In particular, the XML Schema recommendation from the World Wide Web Consortium (W3C) isn't completely or consistently implemented by several vendors. These problems slowed down the development of XBRL 2.1, and they continue to plague developers of XBRL software who are relying on these other vendors. This means that you are getting fewer products later—and at a higher price. *Strategy:* This situation will get better over time, particularly as an improving economic cycle allows money to flow to stalled projects.

HYPERTEXT ● The domain of accounting literature is modeled as a slowly changing hypertext. "Hypertext" is the HT of HTML, the basic technology of the Web. Like the Web, the accounting literature is a set of documents written at different times that are linked by references to each other. These documents are changing slowly, and XBRL taxonomies have to track the changes in accounting standards. The internal policies and procedures of your company can be considered linked to this "hypertext." Developers moving into XBRL with an "accounting software is about numbers" mentality are in for a shock. *Strategy:* Web applications such as Google™ make the Web easier to use. XBRL development also benefits from using middleware, which can conceal the details of this issue from your staff and developers.

CONTROL ● The above-referenced accounting literature hypertext has multiple authors that don't necessarily have control over each other's work, which is produced at different times and places and is available at different locations. (Sound familiar? Now you can tell your management that your monthly consolidation circus is really a "hypertext!") *Strategy:* An application should anticipate dealing with a set of taxonomies, not just one. Conflicts between taxonomies can arise, resulting in either multiple definitions or prohibitions.

ARCHIVAL ● The XBRL taxonomy language is tuned for interchange and maintenance of an "archival" hypertext. It isn't tuned for ease of use inside an application. *Strategy:* This is an unavoidable consequence of the high-level requirements of XBRL alluded to earlier. The result is more work for developers.

ATOMIC ● The XBRL fact language is "tuned" for atomic facts—numbers that can be understood independently, such as, "According to U.S. GAAP, the assets of example.com were exactly $7.46 on December 31, 2003." This is good because the simple should be simple. In addition, the complex should be possible, but XBRL has struggled with how to support facts that depend on each other and facts that are not atomic values. *Strategy:* Don't try to use XBRL where it is weak. XBRL isn't a one-size-fits-all replacement for relational databases or other kinds of XML. For example, XBRL isn't designed to handle operational or transactional data. To support Human Resources applications, look to HR-XML for the

TABLE 1. Sample XBRL Instance for example.com

<xbrl namespaces for XBRL, XML Schema Instance, and US GAAP taxonomy go here>	All XBRL facts are contained inside an <xbrl> tag, and it's a good practice to use this tag to hold the necessary XML namespace declarations.
<link:schemaRef xlink:type="simple" xlink:href=" usfr-ci-2003.xsd"/>	XBRL processors use this reference to find all the taxonomy resources.
<us:assets contextRef="c1" unitRef="u1" precision="INF">7.46</us:assets>	A single business measurement.
<unit id="u1">ISO4217:USD</unit>	The unit of measure is U.S. dollars.
<context id="c1">	The context gathers together details for several facts.
<entity><identifier scheme= "http://www.duns.com/D-U-N-S"> 1234567890</identifier></entity>	The fact applies to the entity identified by a DUNS™ number.
<period><instant>20031231</instant> </period>	The fact applies at the given instant in time.
</context>	The end of the context element.
</xbrl>	The end of the XBRL instance.

right XML language, not XBRL.

PRECISE ● XBRL keeps track of precision and unit of measure for numeric facts. Developers may be unclear on where to find this data when exporting to XBRL or where to put it when importing from XBRL. *Strategy*: Education.

UNITS ● Unit of measure is necessary to ensure "apples-to-apples" comparisons and calculations, as the history of the $125 million Mars Climate Observer demonstrates. Some of the navigation instructions were done using English measures, some using the metric system (from NASA's Office of Space Science, Release 99-113, "Mars Climate Orbiter Team Finds Likely Cause of Loss," September 30, 1999). Developers aren't familiar with keeping track of this and avoiding adding dollars to euros. *Strategy:* More education! The requirement to capture this information for XBRL can help catch errors where other software is less careful about units of measure.

SEPARATION ● XBRL completely separates the business facts from any presentation-related data. Many developers are used to making assumptions—for example, that the physical order of the XML and the containment structure can be used to drive presentation. It's easy for the XBRL instance to contain no presentation hints except the value of the fact itself (7.46 in the above example). See Table 1 for an example. There's nothing in the coding that forces a particular presentation style. *Strategy*: Set some rational expectations for how XBRL will be used and where it will help. The next point expands on this.

INTERCHANGE ● Both the fact and taxonomy languages are data interchange languages. They are designed to be part of larger systems. XBRL helps transfer data between computer applications—that's what it was designed to do best. *Strategy:* While rudimentary presentation effects are possible knowing only what is captured in XBRL, an application trying to style XBRL directly for human consumption has to add a considerable amount. XBRL doesn't know anything about bold face, double underlines, or parentheses around negative numbers.

EMBEDDED ● XBRL facts can be embedded inside other kinds of documents, just like images in a Web page. *Strategy:* Software has to be able to treat each "island" of XBRL independently.

In closing, I want to emphasize that the above are related to *features*, not *bugs*. The points of difficulty can best be addressed with better education (a primer). Free tutorial information is beginning to appear, such as at KPMG's XBRL site (http://www.kpmg.com/xbrl). ■

David vun Kannon is a senior manager in KPMG's Audit and Advisory Services Center.

Neal Hannon is an accounting lecturer for the Barney School of Business at the University of Hartford. Author of two books and numerous articles, he is the IMA's representative to the XBRL International consortium. An elected member of the XBRL-US steering committee, he is the chair of the XBRL-US Education Work Group.