2.5. Decisions, Decisions

Life is full of tradeoffs, and XBRL is no exception. We can sum up the tradeoffs made in
the creation of XBRL in one simple mantra: "Easy stuff should be easy, and hard stuff
should be possible."

Furthermore, XBRL was designed by a committee, which anyone could join. In addition, any
member could participate and push for what he desired. As with any committee-based
design, XBRL isn't perfect. Neither is any other standard. Finally, during the creation of
XBRL, many other Web standards were still in development and were unavailable for use.

Because of the tradeoffs made in the development of XBRL, XBRL has issues that may need
to be overcome in certain situations or use cases. The following sections discuss tradeoffs
made in the development of XBRL and provide insight on how to overcome any negative
impacts.

2.5.1. XBRL uses XML

It may seem like an obvious choice today, but XML was an upstart in 1998 when XBRL got
started. Perhaps you've heard the phrase, "Standing on the shoulders of giants," which was
another mantra of those deciding the architecture of XBRL. The World Wide Web
Consortium (W3C), the giant in this case, was working on a lot of different specifications to
support XML. For example, specifications for encrypting XML or digitally signing an XML
document didn't exist at the time. But everyone anticipated that they would exist, so XBRL
didn't address these issues, but instead planned to leverage what the W3C would
eventually create. As a result, users of XBRL can make use of W3C standards for encrypting
and digitally signing their XBRL.

Standing on the shoulders of giants

Because of the choice to use XML and as many of the W3C specifications as
possible, some things are easier, and some things are harder. If different
choices were made, different things would be harder, and different things
would be easier. With this decision, the right things are easy. For example, XBRL
routinely gets criticized for using XLink to express additional information
(resources) and relations. XBRL could have re-created a linking mechanism
(what XLink already provided), creating a solution that more closely met XBRL’s
need, but the linking approach would be proprietary to XBRL. However, the folks
building XBRL would’ve had to expend resources to figure out all the nuances of
linking things together, basically reinventing what the W3C had created with
XLink.

NOTE

At the time of XBRL 1.0’s creation, how schemas would be expressed was unclear. XBRL 1.0 provided a document-type definition (DTD), but the XBRL creators knew
that XML Schema, or something like it, would exist. As such, the creators of XBRL
didn’t want to re-create an approach to expressing business concepts, but rather
leveraged XML Schema to express business concepts within XBRL. Likewise, rather
than create an XBRL-specific approach to expressing relations, they used XLink
(W3C XML language for creating and describe links) to express such relations.

2.5.2. XBRL is a general-purpose specification

Early in XBRL’s life, before it was even called XBRL, its prototypes were mainly related to
financial reporting. XBRL’s early name was even focused on financial reporting, partially
because those who started XBRL came from that background. But as they thought about it
more and more, those early pioneers realized that what they were trying to create would
have applications far beyond financial reporting. So, they made a choice to focus XBRL not
just on financial reporting, but rather on business information exchange in general.
The world of business information exchange is extraordinarily complex. XBRL had to provide a common baseline for the many types of business information exchange. No one will ever use all the features or characteristics of XBRL at the same time. Most XBRL users won't have all the problems of business information exchange in their use cases for XBRL; they'll have a smaller set of problems. All that XBRL provides may seem excessive for these smaller use cases.

However, XBRL's users will use what they need and ignore the parts that aren't applicable to their situation. This choose-what-you-need approach is one reason XBRL is created in a modular manner.

One impact of this all-purpose nature of XBRL is that, at times, it can be too flexible and have too many options, so systems that make use of XBRL must constrain that flexibility and eliminate undesired options by using an application profile of XBRL (see Chapter 12.)

2.5.3. XBRL uses an atomic approach

One early and unanimous choice in the early development of XBRL was to use an atomic approach to expressing values as opposed to the more constraining content model or document model generally used by XML. The decision's primary driver was flexibility.

To understand this choice, you need to understand the two options:

- **Atomic approach:** This approach models information independently of other information to which it might relate. Separating the information from its relations to create atomic, stand-alone pieces of information is quite flexible. However, understanding information that relates to or is even dependent on other information is more challenging because information isn't bound together like more traditional XML languages, which bind information together using the XML content model.

- **Document approach:** In this approach, you can express information that depends on other information. However, this approach isn't as flexible and makes using information independently harder. This approach leverages the XML content model to bind things in order to keep them together. The content model creates one explicit hierarchical information model. The downside is that separating information for independent use is more challenging.

One of XML's greatest strengths is its hierarchical nature — its ability to express related information in the form of a content model. However, the constraint of a content model has two downsides:

- You can express only one content model, and that content model is implicit. By *implicit*, we mean that you really don't understand if the content model expresses, say, a document, a transaction, or something else. It's simply a content model and what it represents is implied.

- After you express the content model, you can't easily change it so that you can easily communicate those changes to software applications that use that information model. Basically, the document approach's content model was too constraining.

The constraining content model prevented extensibility, which was high on XBRL's priority list. For that reason, the atomic model was the obvious choice.

However, the atomic approach has some downsides. You must create relationships in order to provide human-readable renderings. For example, rendering XBRL into some human-readable formats involves another step, such as generating a usable content model, because most XML tools for rendering rely on the content model to generate the formatting.

**NOTE**

If you feel you need to use the content model when you use XBRL, you can. However, at times, the XBRL's other pieces, which work quite well when you don't use a content model, are harder or impossible to use. For examples, XBRL Dimensions and *tuples*, or compound facts (see Chapter 22), don't work together well; think twice before you attempt to use XBRL Dimensions and tuples together. The impact of the choice to fine-tune XBRL for the atomic approach is that XBRL is quite flexible in specific areas, yet retains constraint in other needed areas.

2.5.4. Reuse is more important than presentation

XBRL separates the notion of expressing business information and the presentation of that business information. The XBRL creators had a hard enough time agreeing on how to
express business information: Agreeing on every nuance of how to present that information would have been impossible.

This separation of the presentation and the information itself means that information users can easily make their own choices about how to present information. The information's creator doesn't dictate how to present that information.

As a result, information can be truly interactive, reformatted on the fly at the whim of the information's users. The downside is that the information's creators may not want users to have that much flexibility in terms of presentation. However, the creators can lock down the presentation. In addition, the separation of presentation and data makes presentation a little bit more difficult for creators and consumers of XBRL.