1.8. Making XBRL Work for You

No standard is perfect, and XBRL is no exception to this rule. If you do choose to make use of XBRL (and you likely will), you should realize the following:

- **XML is a syntax; XBRL expresses semantics.** XBRL goes further than many XML languages in providing the ability to express semantics or meaning. Understanding this difference is critical to understanding XBRL. (Chapter 2 covers this concept in more detail.) Just realize that comparing XML and XBRL isn't a good comparison.

- **XBRL is about creating information structured for meaning.** Currently, many business reports are unstructured, so reusing the information within the report in an automated fashion is impossible. It doesn't have to be that way. Providing information in a format structured for meaning has advantages. Chapter 21 explains the differences between unstructured information, information structured for presentation, and information structured for meaning. Understanding these differences is critical to understanding XBRL.

- **XBRL is a general-purpose specification.** Being general purpose, it has to serve many masters. No one system making use of XBRL will use 100 percent of its features. Rather, users will pick and choose the parts they need, ignoring the parts they don't need.

- **XBRL has no standard logical model.** As a result, you need to follow the physical model of XBRL, which leads to complexity, or create your own logical model, which makes working with XBRL much easier. However, if done incorrectly, this can lead to interoperability issues. You can solve this situation by creating a proper logical model and making use of a well articulated *application profile*, which is a constrained subset of the full general-purpose XBRL specification and specifies your logical model. (Chapter 21 discusses application profiles and logical models.)

- **Specific proprietary solutions are commonly better than standards; standards provide leverage.** Proprietary solutions to a specific problem are commonly better than standards because that is how businesses differentiate themselves — by creating a good solution to a very specific problem. But the thing about proprietary solutions is that they can be expensive, and they lead to the minimum amount of interoperability, which doesn't provide what you really need — a standardized way to describe, exchange, and analyze business information.

- **XBRL taxonomies are data models.** Many people understand the benefits of a well-designed database schema and the ramifications of a poorly designed database schema. A database schema is a data model. An XBRL taxonomy may express a data model and can, and many times should, be treated as such. Just as you can have good and bad data models, you can have good XBRL taxonomies and not-so-good XBRL taxonomies. A bad XBRL taxonomy model leads to poor interoperability. Poor interoperability leads to people needing to be involved in information exchanges. When you consider your XBRL taxonomy data model design, you need to consider the extensions others will be creating. All these issues are best dealt with by using an information modeling layer, which can help you create good data models and thus good XBRL taxonomies. (Chapters 12 and 17 discuss information models.)

- **XBRL taxonomies can have different architectures.** XBRL is a general-purpose specification. You can choose from several ways to model your XBRL taxonomies, which dictate what your XBRL instances will look like and how they act. Many times, different architectures aren't as compatible as you might hope. This incompatibility may result in interoperability issues between different XBRL taxonomies, which is important to realize because, in all likelihood, your organization will be dealing with many different XBRL taxonomies. For example, you can report to different regulators, which use different XBRL taxonomies. Further, you may choose to make use of XBRL internally within your organization. Managing the interoperability of your internal business systems and various XBRL taxonomies architectured in different ways can be a challenge. One way to deal with these realities is to create an abstraction layer between your internal implementation of XBRL and other implementations of XBRL, which helps minimize the impact of the whims of others on your important internal
business systems. (Chapter 12 discusses how to maintain control by using an abstraction layer.)

- **XBRL is not a complete solution.** XBRL is a technology that provides significant leverage, and you can use it within a system. You can string together all the pieces you need, or you can adopt an architecture, an application profile, and an information model created by others. What you can’t do is simply pick up XBRL, plug it in, and expect all your problems to be solved.

- **An island of XBRL isn't an effective goal.** Creating an island of XBRL within an organization doesn’t serve any real purpose. All that approach does is create additional work and additional cost with no true marginal benefit to the system as a whole.

At first glance, you may not have considered the preceding aspects, the realities, of working with XBRL. But eventually, you’ll run up against these issues.

*Chapter 2* helps you become well grounded in the realities of working with XBRL. Considering these realities as you figure out how to apply XBRL within your organization can help you minimize false starts, point out dead-end paths, and otherwise learn from the missteps of others so that you don’t make the same mistakes.