

# Understanding and Using XBRL Extensibility

XBRL has the term "extensibility" within its name (as does XML). By understanding the different approaches to extensibility users of XBRL can make better design choices, ease implementation, and improve the quality of their XBRL projects. In this section we explain several extensibility mechanisms, the best uses of each type of approach, and help users of XBRL pick the right mechanism for the right task.

## 1.1. Terminology

To begin, we want to get a firm grounding for a few terms used in this document to be sure we are on the same page.

### 1.1.1. Portability

"Portability" is the first term we want to define. Portability, as used here, has to do with taking something from one environment and using it within another environment. Portability also is used to describe how easy it is to move something between two platforms (e.g., across operating systems) or applications (e.g., exchanging data or metadata between software applications on similar or different platforms).

Here are some examples of portability.

Imagine that an application is created and the business rules used for that application are stored in the application itself. Those business rules are not reusable outside that application. This is an example of something not portable. But what if the business rules are expressed and stored outside the application? In that case, another application could understand those rules. Then, if the two applications exchanged data they could also exchange the business rules which go along with the data.

Another example of portability might be a database schema which is defined outside the database, and therefore could be exchanged.

Basically, something is portable if it is independent of the application in which it was created. Portability is increased if the data model is well designed, and the data representation technique well documented. The rule of thumb is this. If you want to make it easier to reuse your data, it's best to invest up front to make it portable.

### 1.1.2. Leverage

We use the term "leverage" to indicate the amount by which something created for one purpose can be re-purposed for something else. Something provides a high degree of leverage if it makes using it for something else easier, thus reducing time, reducing implementation costs, decreasing training costs, etc.

For example, open standards provide leverage. While an open standard may not provide a perfect solution to a specific problem, the open standard may provide a solid starting point which can be built upon, therefore reducing the total cost of a solution.

## 1.2. Summary of Extensibility Methods

The following table summarizes various methods of extending the mechanisms for representing data and data relationships. Each method is detailed in later sections.

<b>Method of Extensibility</b>	<b>Portability</b>	<b>Explanation</b>
XML Extensibility	Not really portable in a general way	XML extensibility is provided by virtue of the XML specification. It's strength is its flexibility. Its weakness also is also its flexibility. You generally can do anything XML allows without breaking anything, but you won't have a wide audience for your extensions unless it gets widely adopted. An example of this is the attributes added by the XBRL Dimensions 1.0 Specification.
XML Schema Extensibility	Generally portable. However, some extensions won't behave as expected. For example, substitutionGroup inheritance is <i>not</i> respected following relationships.	Anything from XML Schema, Part 2: Datatypes ( <a href="http://www.w3.org/TR/xmlschema-2/">http://www.w3.org/TR/xmlschema-2/</a> ), can be used within XBRL. For example, restrictions, custom types, substitutionGroups, etc. An example of this is adding an enumerated list to a type.
Basic XBRL Extensibility	Portable	Basic XBRL Extensibility is defined as the basic stuff XBRL allows you to do: create taxonomies, add concepts, create presentation, calculation, definition relations, and add label and reference resources. Basic, basic stuff. You do this, any XBRL application can process what you create. An example of this is any XBRL taxonomy.
LRR Based XBRL Extensibility	Portable to the extent an application understands the LRR roles, arc roles, etc.	This type of extensibility relates to adding extended link roles, arc roles, and resource roles and defining meaning to those roles. Again, this fits nicely into XBRL applications. An example of this is the arc roles defined by the XBRL Dimensions specification.
Custom Extensions Using XBRL Philosophies	Not generally portable, only to the extent applications support reading and using the custom linkbase. However, the portable (linkbase) syntax gives others an opportunity to support the extension.	This type of extensibility adds relations or resources to XBRL using custom linkbases created in a manner compatible with the XBRL linkbases. An example of this is the UBmatrix Formulas Linkbase.

<b>Method of Extensibility</b>	<b>Portability</b>	<b>Explanation</b>
Proprietary Extensions using Whatever Means	Not portable	As the components of XBRL are exposed, consistent, and generally usable by applications, other applications can access this information from whatever source. Examples might be a relational database connected to XBRL in some way, or business rules coded directly into an application.

Generally speaking, portability is based on a community agreeing on syntax and semantics. For example, if XBRL vendors implement something in the Link Role Registry (LRR), then it will be more portable. One of the major benefits of XBRL is that it provides mechanisms which can be used to forge such agreement. Examples of this are the modules of XBRL such as the XBRL Dimensions specification, and extension mechanisms such as the LRR. Other XBRL extension mechanisms are likely. For example, there is an expressed desire to create an XBRL Functions registry, similar to the LRR, but for functions used in the XBRL Formulas specification.

### 1.3. Details of XBRL and XML Extensibility Methods

In this section we will drill down into the XBRL extensibility mechanisms outlined in the previous table. We point out the pros and cons of each.

#### 1.3.1. XML Extensibility

XBRL is XML and therefore leverages all the features of XML. One of the best features of XML is its extensibility. When we talk about XML we mean XML itself, XML Namespaces, and the entire body XML-related technologies (e.g., XLST, XPath, XLink).

One example of XML extensibility which is usable within XBRL is simply adding an attribute to an element within an instance document or a taxonomy. Consider this example:

```
<ci:Director id="ID-01" code="Test">
  <ci:Name contextRef="D-2003">Ho Ching</ci:Name>
  <ci:Salary contextRef="D-2003" unitRef="U-Monetary" decimals="INF">0</ci:Salary>
  <ci:Bonus contextRef="D-2003" unitRef="U-Monetary" decimals="INF">0</ci:Bonus>
  <ci:DirectorFees contextRef="D-2003" unitRef="U-Monetary" decimals="INF">60000</ci:DirectorFees>
  <ci:FairValueOptionsGranted contextRef="D-2003" unitRef="U-Monetary"
decimals="INF">0</ci:FairValueOptionsGranted>
</ci:Director>
<ci:Director id="ID-02" code="Test">
  <ci:Name contextRef="D-2003">Boon Swan Fo</ci:Name>
  <ci:Salary contextRef="D-2003" unitRef="U-Monetary" decimals="INF">879639</ci:Salary>
  <ci:Bonus contextRef="D-2003" unitRef="U-Monetary" decimals="INF">1213486</ci:Bonus>
  <ci:DirectorFees contextRef="D-2003" unitRef="U-Monetary" decimals="INF">0</ci:DirectorFees>
  <ci:FairValueOptionsGranted contextRef="D-2003" unitRef="U-Monetary"
decimals="INF">569000</ci:FairValueOptionsGranted>
</ci:Director>
```

Note the "id" and "code" attributes on the "ci:Director" tuple. Neither of these attributes are XBRL attributes. They are added within an XBRL taxonomy, which is a valid XML Schema. Now, the "id" attribute may not look like an XML attribute but it is. (By the way, it is recommended by FRTA that every taxonomy add this attribute as it was a oversight of the XBRL specification to omit allowing XML ID attributes on tuples.) But to make this more clear we added a second attribute to the tuple, the "code" attribute. Attributes are one way where XML Schema extensibility can be

used. If you load this taxonomy or instance document into an XBRL application the instance document does not "break" the application.

This "not breaking the application" is a very useful feature. But, it also is the case that if the XBRL application does not know what to do with the attribute or element added it may not understand how to use it. For internal extensions to XBRL where you have control over all users, this attribute extension is a very useful feature. But, if you were to make these attributes available more generally, it is likely that they will not be utilized.

A second example of XML extensibility relates to the "segment" and "scenario" elements within an XML instance document. Consider the following instance document fragment.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created by Charles Hoffman, CPA, UBmatrix: 2005-07-07 -->
<xbrl xmlns="http://www.xbrl.org/2003/instance"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:ci="http://www.UBmatrix.com/Patterns/ComplexConcept"
  xmlns:scenarios="http://www.xbrl.org/frta/scenarios"
  xmlns:segments="http://www.xbrl.org/frta/segments"
  xmlns:iso4217="http://www.xbrl.org/2003/iso4217"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.UBmatrix.com/Patterns/ComplexConcept ComplexConcept.xsd
    http://www.xbrl.org/frta/scenarios scenarios.xsd
    http://www.xbrl.org/frta/segments segments.xsd"
  ">

  <link:schemaRef xlink:type="simple" xlink:href="ComplexConcept.xsd" />

  <context id="D-2003">
    <entity>
      <identifier scheme="http://www.SampleCompany.com">SAMP</identifier>
      <segment>
        <segments:ReportingSegment><segments:Group /></segments:ReportingSegment>
      </segment>
    </entity>
    <period>
      <startDate>2003-01-01</startDate>
      <endDate>2003-12-31</endDate>
    </period>
    <scenario>
      <scenarios:ReportingScenario><scenarios:Actual /></scenarios:ReportingScenario>
    </scenario>
  </context>
```

Note the <segment> and <scenario> elements within the context above. The XBRL specification allows basically any XML within these elements. We have created two XML Schemas in the files "segments.xsd" and "scenarios.xsd" and provided the elements which we would use within our <segment> and <scenario> elements. (See the sample subdirectory for this section.)

This use of XML within the <segment> and <scenario> elements points out the strength and weakness of using XML. We can provide this XML but what does it mean? How are we to interpret the information provided in these elements?

This literally unlimited flexibility, and understanding what the creator of the information intends, can be overcome by creating a specification, explaining what is intended, and then getting others to adopt that specification. We will cover this approach more as we cover other extensibility options.

As you can see XML extensibility may work well in some cases and poorly in others. XML extensibility definitely is useful but not always the best choice.

### 1.3.2. XML Schema Extensibility

One of the things that may people do not realize is that it can use literally anything within XML Schema Part 2: Datatypes. This can be particularly useful in constraining the values allowed within instance documents. For example, one may use the XML Schema features for adding facets to a data type. These facets can be used to constrain data such as the length of a string, items from an enumerated list, a range of numbers between some minimum and maximum value, etc.

For example,

```
<complexType name="textItemType">
  <simpleContent>
    <restriction base="xbrli:normalizedStringItemType" />
  </simpleContent>
</complexType>
<complexType name="PercentItemType">
  <simpleContent>
    <restriction base="xbrli:decimalItemType">
      <minInclusive value="0" />
      <maxInclusive value="1" />
    </restriction>
  </simpleContent>
</complexType>
<complexType name="SegmentTypeItemType">
  <simpleContent>
    <restriction base="xbrli:stringItemType">
      <enumeration value="Business" />
      <enumeration value="Geographic" />
    </restriction>
  </simpleContent>
</complexType>
```

In the above code fragment, three data types are defined which can be used as a type for any XBRL concept within a taxonomy. These are defined below:

- The complexType "textItemType" is created to use a "normalizedStringItemType" as it's base type. It simply provides a more "user friendly" name for a data type. Its properties are simply those of the XML Schema type "normalizedString" as originally modified to be an XBRL item by XBRL.
- The complexType "PercentItemType" defines a data type used for expressing percentages – a type not provided by XML Schema. It says that the value must be between 0 and 1. This clarifies how a percentage should be provided within a fact value; showing that 10 percent should be entered as not "10", but rather as ".10".
- The complexType "SegmentTypeItemType" defines an enumerated list of the two different types of segments: "Business" and "Geographic".

Even with just these three examples one can see that facets are a very powerful feature of XML Schema also fully available within XBRL.

A second very useful feature of XML Schema is the "substitutionGroup" attribute. XBRL makes heavy use of substitution groups. In fact, every concept within XBRL is in one of three substitution groups: xbrli:item, xbrli:tuple and xbrli:referencePart. Examining the use of these within XBRL can provide good insights as to how substitution groups can be used. Consider the following example

```

<!-- Definition of the complex type which is used as a "template" for all PPE classes -->
<complexType name="ClassOfPropertyPlantEquipmentComplexType">
  <!-- This is for depreciable PPE, may create another complexType for non-depreciable PPE -->
  <complexContent>
    <restriction base="anyType">
      <sequence minOccurs="0" maxOccurs="1">
        <element ref="ci:NameClassPropertyPlantEquipment" />
        <element ref="ci:CodeClassPropertyPlantEquipment" />
        <element ref="ci:MeasurementBasisForClassPropertyPlantEquipment" />
        <element ref="ci:DepreciationMethodForClassPropertyPlantEquipment" />
        <element ref="ci:EstimatedUsefulLifeForClassPropertyPlantEquipment" />
        <element ref="ci:AmountClassPropertyPlantEquipmentUnderFinanceLeases" />
        <element ref="ci:ProFormaAmountClassPropertyPlantEquipment" />
        <element ref="ci:MovementsInClassPropertyPlantEquipment" />
        <element ref="ci:AmountClassPropertyPlantEquipment" />
        <element ref="ci:AmountAdditionsForClassPropertyPlantEquipment" />
        <element ref="ci:AmountDisposalsForClassPropertyPlantEquipment" />
        <element ref="ci:AmountTranslationDifferenceForClassPropertyPlantEquipment" />
        <element ref="ci:AmountTotalChangeForClassPropertyPlantEquipment" />
      </sequence>
      <attribute name="id" type="ID" use="optional" />
    </restriction>
  </complexContent>
</complexType>

<element abstract="true" id="ci:AbstractPropertyPlantEquipmentClass"
name="AbstractPropertyPlantEquipmentClass" type="ci:ClassOfPropertyPlantEquipmentComplexType"
substitutionGroup="xbrli:tuple" nillable="true" />

<element id="ci:LandBuildings" name="LandBuildings"
type="ci:ClassOfPropertyPlantEquipmentComplexType"
substitutionGroup="ci:AbstractPropertyPlantEquipmentClass" nillable="true" />

<element id="ci:FurnitureFixtures" name="FurnitureFixtures"
type="ci:ClassOfPropertyPlantEquipmentComplexType"
substitutionGroup="ci:AbstractPropertyPlantEquipmentClass" nillable="true" />

<element id="ci:Other" name="Other" type="ci:ClassOfPropertyPlantEquipmentComplexType"
substitutionGroup="ci:AbstractPropertyPlantEquipmentClass" nillable="true" />

```

The above fragment of XML (simplified for easier reading and understanding) shows the creation of a reusable "ClassOfPropertyPlantEquipmentComplex-Type" which can be reused, making creation of new classes of the concept easier for users. All a user needs to do is create one concept in a taxonomy and they have an entire, and consistent, complex type; simply by using substitution groups.

A detailed discussion of this is beyond the scope of this material. See the "Modelling Financial Reporting Patterns" section, within the sub-section "Class Properties" for a complete set of examples.

An XBRL processor must be able to validate all these XML Schema features. In fact, XBRL processors do understand XML Schema. However, XML Schema processors do not understand all features of XBRL.

### 1.3.3. Basic XBRL Extensibility

Basic XBRL extensibility is defined as using the base features of XBRL: creating concepts, adding label or reference resources, or expressing calculation, presentation or definition relations.

XBRL provides "boundaries" for this extensibility. Another way of saying this is that XBRL's extensibility is prescriptive. If you stay within those boundaries, an XBRL processor or other application which understands XBRL will understand your extension.

The XBRL specification defines these extension mechanisms, which any XBRL processor understands. An XBRL processor can take a base set of concepts and relations, take additional concepts, resources, or relations and meld them into a new combined set, called a discoverable taxonomy set (DTS). Related to this are mechanisms for removing or prohibiting existing relations or resources, effectively removing them from the DTS.

Examples of basic XBRL extensibility provided, which any application which has an XBRL processor will understand are things like:

- Adding a new XBRL concept to a schema.
- Adding a new label or reference for a concept.
- Adding a new presentation, calculation, or definition linkbase to express additional relationships.
- Modifying a presentation, calculation, or definition relationship (basically prohibiting the existing relation and creating a new relation).
- Adding a new extended link role which can be used to define a new network of relations within a linkbase.
- Adding a new arc role or resource role which can be used to define a relation within a linkbase.

All the above modifications are done without impacting the original set of schemas or linkbases. All the new concepts and relations are built in separate physical files, reference the existing physical files (which you likely don't even have the ability to modify), creating a new DTS for your purposes. And an XBRL processor understands how to put all the pieces together, if you stayed within the boundaries of XBRL. If you did not stay within the boundaries, the XBRL processor will simply ignore the modifications, such as an XML attribute added for a proprietary purpose.

Fundamentally, the XBRL specification is the way to express taxonomies; not any one specific taxonomy. All taxonomies are extensions of XBRL, built on top of the XBRL specification. The basic XBRL specification and additional XBRL modules provide this base extensibility. A conformant XBRL processor understands all the rules relating to this. It is just basic XBRL.

#### **1.3.4. Link Role Registry (LRR) Based XBRL Extensibility**

LRR based XBRL extensibility adds new functionality not provided by XBRL itself. This functionality is added in ways and using methods which XBRL does understand. (Note: This particular feature is described in even greater detail in section 1.4.)

The LRR is a specification of XBRL International. What is important to understand about the LRR at this point is that it is a formal mechanism for defining extended link roles, arc roles and resource roles. Part of this method involves (a) defining the new functionality, (b) creating conformance tests to help articulate precisely how this functionality will operate, and (c) sharing this specification with others so that they do not have to reinvent it.

Two examples will help see what the LRR helps achieve. The first example is the "restated label" label role defined by the IFRS-GP taxonomy. The XBRL specification defines a number of label roles which can be used by taxonomies. One commonly used role, but discovered after the release of the XBRL 2.1 Specification, is the "restated label". This kind of label is used if a value is restated. For example, an "normal" label may be "Retained Earnings" or perhaps "Retained Earnings, Originally

Stated". Prior period adjustments might be shown then a new balance for retained earnings of "Retained Earnings, Restated". This is when a restated label might be used.

This is the URL where the restated label role is defined:

<http://xbrl.iasb.org/int/fr/ifrs/gp/2005-05-15/restatedLabel.xsd>

This is defined a second time in a newer version of the IFRS-GP taxonomy:

<http://xbrl.iasb.org/int/fr/ifrs/gp/2006-08-15/restatedLabel.xsd>

Rather than the IFRS-GP taxonomy defining this label, it will eventually end up in the LRR so that any other taxonomy, such as the USFRTF, can also make use of that label role.

A second example of using the LRR is the "aggregator-contributor" role used in calculations relating to dimensional information. The XBRL Dimensions specification does not specify how calculations work. It was consciously left out of the specification for two reasons: (a) there were too many details to work out and it would have delayed the base specification, (b) it could be added as a module.

As such, a separate specification was created to articulate how calculations were to work for dimensions, further leveraging XBRL's ability to express calculations. However, dimensional calculations are cross context, and XBRL calculations were always in the same context. So, rather than using the "summation-item" calculation arc role, a new arc role was defined (rather, for conceptual clarity, *had* to be defined). The schema defining that role is located at the following URL:

<http://www.ubmatrix.com/XBRL/xbrldta-2005.xsd>

This is the calculation arc role which is defined in that schema:

<http://xbrl.org/int/dim/arcrole/aggregator-contributor>

This is not an official LRR entry at this point. It likely will be. Note that other vendors can mimic the "official LRR" with proprietary LRR-type systems for specifying functionality. You can even make such extensions, too. the technology is there.

For the aggregator-contributor, a specification was written and conformance tests created. Vendors can implement the functionality and test their applications against the conformance suite. This is one way to promote the transition of proprietary functionality into shared functionality.

We are pointing out several things here. First, the LRR is a mechanism for creating additional functionality. It may not be appropriate for everything but there are many things it is appropriate for. Second, the LRR is a literal location, an "official" location to put these functionality extensions. Third, this same mechanism can be used for specifying proprietary functionality, say within an organization. XBRL processors understand and can work with this mechanism.

### **1.3.5. Custom Extensions using XBRL Philosophies**

Another approach to extensibility is to create custom extensions, but follow XBRL philosophies. An example of this is the UBmatrix Formulas Specification.

The XBRL specification allows for custom linkbases, provided they are created per the XBRL specification. The XBRL 2.1 conformance suite has within it tests which these custom linkbases must pass and, therefore, as long as custom linkbases are

created in the appropriate manner an XBRL processor will be able to read these linkbases. Now, the XBRL processor may not know what to do with these linkbases as it will not "know" how to process the linkbase. But, the custom linkbase will not break an application, and can ignore the added value provided by the custom linkbase, instead falling back and processing what it does understand. In this case, the "not breaking the application" features is very useful.

The following URL shows a schema created for a custom linkbase which is used to create formulas, or business rules:

<http://www.ubmatrix.com/XBRL/ubm-xbrl-linkbase-formula-2006-06-15.xsd>

The following URL identifies a custom formula linkbase which can be processed by an XBRL processor, and which validates per the schema above:

<http://public.xbreeze.ubmatrix.com/OpenSource-IFRS/IFRS-Basic-Rules/2006-07-15/ifrs-basic-rules-2006-07-15-formula.xml>

It is beyond the scope of this book to provide a detailed explanation of exactly how to create a schema for a custom linkbase and to create and use the custom linkbase within an application. However, we want to provide an example of a use case and show a high level view of using custom linkbases and point out that they can be created.

The UBmatrix formulas linkbase was created for two reasons. First, because XBRL International had not provided a linkbase for expressing business rules or formulas. Such a linkbase is currently being created, but it is not available for use at this point. The second reason is that FDIC (Federal Deposit Insurance Corporation) needed to express approximately 1800 business rules within a proprietary production system making use of XBRL. As such, this linkbase was created. It follows the philosophies of XBRL and can be validated by fully compliant XBRL processors. However, the processor will not be able to process the formula resources provided in the linkbase unless the application fully supports the custom linkbase.

To give you a sense of what the custom linkbase provides, let's look at the UBmatrix formulas linkbase:

```
<?xml version="1.0" encoding="utf-8"?>
<link:linkbase
  xmlns="http://www.ubmatrix.com/2004/XLink/xbrllinkbase/formula"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xsi:schemaLocation="
    http://www.ubmatrix.com/2004/XLink/xbrllinkbase/formula
    http://www.ubmatrix.com/XBRL/ubm-xbrl-linkbase-formula-2006-06-15.xsd
  ">
  <link:arcroleRef xlink:type="simple" xlink:href="http://www.ubmatrix.com/XBRL/ubm-xbrl-linkbase-
  formula-2006-06-15.xsd#concept-formula" arcroleURI="http://www.UBmatrix.com/2003/arcrole/concept-
  formula" />
  <link:roleRef xlink:type="simple" xlink:href="http://www.ubmatrix.com/XBRL/ubm-xbrl-linkbase-
  formula-2006-06-15.xsd#formula" roleURI="http://www.UBmatrix.com/2003/role/formula" />
  <formulaLink xlink:type="extended" xlink:role="http://www.xbrl.org/2003/role/link">
    <link:loc xlink:type="locator" xlink:href="gaap-Rules.xsd#gaap-rules LandBuildingsReconciles"
  xlink:label="gaap-rules LandBuildingsReconciles" />
    <formulaArc xlink:type="arc" xlink:arcrole="http://www.UBmatrix.com/2003/arcrole/concept-
  formula" xlink:from="gaap-rules LandBuildingsReconciles" xlink:to="gaap-
  rules LandBuildingsReconciles formula" />
    <formula xlink:type="resource" xlink:role="http://www.UBmatrix.com/2003/role/formula"
  xlink:label="gaap-rules LandBuildingsReconciles formula" select="gaap:LandBuildings[-PIY] +
  gaap:ChangesTotalLandBuildings = gaap:LandBuildings" />
  </formulaLink>
</link:linkbase>
```

Notice the following from the XML above:

- Note the reference to the schema file in the `xsi:schemaRef` attribute
- Notice the `<formulaLink>` element
- Notice the `select` attribute which contains the actual formula

Now, you may not understand exactly how the formula linkbase above actually works, but that is not the point. The point is that if you have something you wish to express as a custom linkbase, there are ways for you to do so, leveraging a lot of what XBRL (and XML, XLink, etc) provides, saving yourself some work, and making what you create work within XBRL applications and not break them. Now, that's the way to create additional XBRL functionality!

We want to remind you that it was not our intention to show you precisely how to create custom linkbases, but rather to show you it can be done. The samples provided can help you reverse engineer functionality you may desire to create. An IT professional will know how to obtain the information they need in order to create custom extensions to XBRL.

### **1.3.6. Proprietary Extensions Using Whatever Means**

And finally, there is one last approach to extending the functionality of XBRL: do whatever you like! While the result of what you create might not be portable to other environments, if they only need to work within your environment, you can pretty much do whatever you need to do.

XBRL is XML. Simply having the information available in a structured format, rather than unstructured, facilitates using the data and extending functionality. While it is challenging, if not impossible, to reuse information contained in, say, Microsoft Word, the possibilities are endless with XBRL and XML. This is particularly true because of the broad global adoption of XBRL and XML.

XBRL is particularly suited for working within relational database management systems (RDBMS). The "flat" structure of XBRL instance documents make it well-suited for conventional database type systems.

An example of a proprietary extension to XBRL would be, say, the Federal Depository Insurance Corporation (FDIC) creating and sorting additional information related to its taxonomy which would not fit into XBRL within its database applications.

## **1.4. Link Role Registry (LRR)**

We have referred to the XBRL Link Role Registry Specification (LRR). Now we want to explain what it is in more detail.

The LRR is a specification of XBRL International. See <http://www.xbrl.org/LRR/>. The LRR provides information relating to the purpose, usage, intended impact on XBRL validation, and other such information on additional extended link roles, arc roles, and resource roles which are used in XBRL taxonomies. Specifically, the LRR is an agreed upon extension mechanism. This need for extensibility is so strong that there is a technical *and* a procedural way to extend XBRL quickly and easily, relative to other approaches.

The benefit of the LRR is leverage. The LRR provides a means of communicating how software should act. To that end it has a mechanism for expressing conformance suites to ensure the software acts in the specified manner.

The following is an example of an LRR entry for a role:

```
<role>
  <roleURI>http://www.xbrl.org/2005/role/restatedLabel</roleURI>
  <status>REC</status>
  <versionDate>2005-05-05</versionDate>
  <authoritativeHref>lrr-types-2006-04-26.xsd#restatedLabel</authoritativeHref>
  <requirement xml:lang="en">FRTA label for restated.</requirement>
  <definition xml:lang="en">This is a recommended role.</definition>
  <versionOfXBRL>2.1</versionOfXBRL>
  <minimumEditionDate>2003-10-22</minimumEditionDate>
  <impactsTaxonomyValidation>>false</impactsTaxonomyValidation>
  <impactsInstanceValidation>>false</impactsInstanceValidation>
</role>
```

The role above defines a label role which can be used on restated labels. This role is not provided by the XBRL specification, is in use in many XBRL jurisdictions, and is therefore provided in the LRR.

While it is the case that any taxonomy creator can define such roles when they create taxonomies, having roles at this higher LRR level provides leverage.

This URL is the physical location of the actual LRR:

<http://www.xbrl.org/lrr/lrr.xml>

The LRR is empty at the current time.

## 1.5. XBRL Modularity/Components

XBRL is modular. Sometime people looking at XBRL don't realize that you can pick and choose the parts you need, adding only the components you need or adding components little by little, rather than building out your solution all at once. There is another reason for looking at XBRL from the perspective of modularity. That is to see more clearly what each component provides in terms of features.

We now look at XBRL from both of these perspectives. We will walk through a fairly simple example, adding XBRL, components piece by piece, showing the incremental functionality provided by each new building block and how it can be done.

We will not provide the intimate details here, but rather share enough information where you can use a set of XBRL files, look at them, and see some important ideas about modularity.

The discussion below is intended for someone familiar with XML and XML Schema, not for an XML novice. The intent is to help you understand XBRL modularity, not teach XML or XML Schema.

### 1.5.1. Starting with an XML Schema

Let's start with only an XML Schema. See the schema below:

```
<?xml version="1.0" encoding="utf-8"?>
<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xbrli="http://www.xbrl.org/2003/instance"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:ci="http://www.UBmatrix.com/Patterns/BasicCalculation"
  targetNamespace="http://www.UBmatrix.com/Patterns/BasicCalculation"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
```

```
<import namespace="http://www.xbrl.org/2003/instance" schemaLocation="xbrl-instance-2003-12-31.xsd"
/>

<element name="PropertyPlantEquipment" substitutionGroup="xbrli:item" xbrli:periodType="instant"
abstract="true" type="xbrli:stringItemType" />
<element name="Building" type="xbrli:monetaryItemType" substitutionGroup="xbrli:item"
xbrli:periodType="instant" />
<element name="ComputerEquipment" type="xbrli:monetaryItemType" substitutionGroup="xbrli:item"
xbrli:periodType="instant" />
<element name="FurnitureFixtures" type="xbrli:monetaryItemType" substitutionGroup="xbrli:item"
xbrli:periodType="instant" />
<element name="Land" type="xbrli:monetaryItemType" substitutionGroup="xbrli:item"
xbrli:periodType="instant" />
<element name="Other" type="xbrli:monetaryItemType" substitutionGroup="xbrli:item"
xbrli:periodType="instant" />
<element name="TotalPropertyPlantEquipment" type="xbrli:monetaryItemType"
substitutionGroup="xbrli:item" xbrli:periodType="instant" />
</schema>
```

The schema above is as about as simple as XBRL gets. As all elements in XBRL are required to have substitution groups, and as all elements with substitution groups are required to be global, XBRL is rather "flat", meaning there is not a lot of nesting of concepts.

The schema above can be validated against the XBRL schemas using any XML Schema validator. However, the XML Schema validator will not perform other validation of syntax and semantics articulated in the XBRL Specification; only XBRL validators, which are familiar with XBRL, can do that additional validation.

With only the schema file it is probably easy to see what you get, if you understand XML Schema. You also should notice what is it that you don't get, particularly the following things desired by business users:

- You don't have user friendly labels in perhaps multiple languages.
- You have no documentation or references which explain what the concepts mean in a standard way.
- You have no way to organize and present the concepts in ways more useful to humans, such as showing concepts in a particular order.
- You have no idea of the computations which exist, the numeric relations between the concepts.
- You have no standard way of adding additional information such as other properties of the concepts.
- You have no way to express other business rules.
- And you do not know if the values for the concepts are to be "debits" or "credits".

All these things above, which business users tend to desire, can be added using a prescribed approach with XBRL. You can add any of these things yourself using XML or some other means, but you have to first invent that approach and then articulate the information you choose to articulate. And if you want to share that approach with others you have to write a specification so other software tools can understand precisely what you intend. If you want to get more and more people to use your approach, you have to convince them that your approach is the best approach, i.e. get global agreement.

XBRL has already done all these things. And if you have an even longer list of things you want to be able to do, you can even add those things using mechanisms provided by XBRL.

### **1.5.2. Adding User Friendly Labels in Multiple Languages**

For purposes of example, assume you have implemented only the XML Schema portion of XBRL. Now, users are complaining that they would like three things:

1. Human readable labels
2. The ability to have those labels in multiple languages
3. And the ability to specify specific types of labels which can be used in different circumstances, for example if a value is positive use label "a" and if a value is negative, then use label "b"

The XBRL label linkbase meets all these requirements. To use the label linkbase, simply create it as specified by the XBRL specification and other XBRL software will understand it and take advantage of it.

### **1.5.3. Adding Documentation or Instructions**

The label linkbase cannot only be used for labels, but rather for any textual-type information (e.g., documentation or instructions). When the Federal Deposit Insurance Corporation (FDIC, see "Business Case for XBRL" in a previous section) implemented XBRL within their systems, they took their documentation, called instructions, and put it into an XBRL label linkbase. They used XHTML, which the label linkbase allows as content for labels. Previously the instructions had been made available in a variety of formats including Word and PDF. Now, because the information is standardized in XBRL, software vendors can automate updates to software applications, rather than cutting and pasting from Word or PDF.

Although not required, you can see how user documentation can be added to XBRL concepts.

### **1.5.4. Adding References to External Documentation**

Another module which is somewhat similar to the label linkbase, in that it is a resource-type linkbase, is the reference linkbase. The reference linkbase allows a taxonomy creator to reference external documentation (e.g., and existing set of instructions or documentation in HTML, PDF, Word, XML, or other format). The reference linkbase allows a user to use pre-existing sets of "reference parts," or to create their own reference parts. Then, information about specific reference information, including how to get to that information via a URL or some other means, can be provided within an XBRL taxonomy.

### **1.5.5. Adding User Friendly Presentation Information**

While XML provides a content model which can then be used to help stylesheets render XML Schema information to users, XBRL tends to be quite flat and therefore more challenging to format using XSLT. Yes, there is a disadvantage of the XML content model: you get only one content model.

With XBRL, adding a presentation linkbase to your schema allows taxonomy creators not only to provide information which can be used to provide human readable information using XSLT, but also provide multiple different options for styling

information. We are not talking about multiple style sheets, which you can also do with XML, but rather multiple content models (which you cannot do in XML). As XBRL is normalized in this regard, pulling the content model out of the XSD and putting the formatting information in a separate presentation linkbase, you can provide not only one presentation linkbase, you can provide any number of presentation linkbases you desire.

Adding one or more presentation linkbases provides taxonomy creators a way of ordering concepts in a particular order, organizing parent-child hierarchies in the form of a tree or graph (graph in the IT sense, not a chart), and you can even specify which label role you would prefer to use for a specific relation. All this is achieved by adding an XBRL presentation linkbase to your XSD.

### **1.5.6. Adding Calculations**

XBRL provides a simple means of expressing simple arithmetic relations or simple allocations, the calculation linkbase. In our case, we have a set of numbers which add up: "Total Property, Plant and Equipment = Land + Buildings + Furniture and Fixtures + Computer Equipment + Other". This can easily be expressed using XBRL calculation linkbases, and you can make that functionality available by simply creating such a linkbase.

### **1.5.7. Adding XBRL Dimensions**

If cross-context calculations or if other cross-context information needs to be expressed, add XBRL Dimensions. For example, expressing business or geographic segmental breakdowns and the relationships between the segments.

### **1.5.8. Adding Formulas (Business Rules)**

If more complex relations exist, then formulas, or business rules, can be added via the use of an XBRL Formulas linkbase.

### **1.5.9. Adding Definitions**

At times, it is quite useful to add additional properties or other information to an XML Schema element. For example, in our case, it is helpful to distinguish "Depreciable" from "Non-Depreciable" schema elements.

This is achieved in XBRL by creating and attaching a definition linkbase to your XSD. With a definition linkbase, literally any amount of additional information can be provided.

### **1.5.10. Adding Other Information**

Still have something you want to associate with your XML Schema elements (such as the this section started with)? An easy way to do this, if you have invested in an XBRL processor, is to create your own custom linkbase to contain whatever information you desire to express. Just follow some simple (for an IT person) rules, and you won't break an XBRL processor, which will simply either (a) ignore what you provide or (b) implement what you provided, if they see fit to do so. See section 1.3.5)

## **1.6. XBRL and Other XML Standards**

XBRL uses many other XML standards such as XML Schema, XLink, and XHTML to achieve things XBRL needs to achieve. For example, XBRL labels may contain XHTML from a specific set of XHTML modules.

It can be more challenging to use other XML standards within an XBRL instance document. For example, XBRL items may not contain any other mark-up. It is likely that at some point XBRL items may be allowed to contain mark-up, for example an XBRL item which has a data type of XHTML, or some specific XML Schema.

On the other hand, XBRL can be imbedded quite easily within other forms of XML. For example, RIXML allows XBRL to be embedded, as does NEWSML.

[CSH: Which standards do this needs to be verified.]

## **1.7. Summary**

XBRL is built to be extensible in a prescribed way (if you care about others using your extensions), or to your hearts content in any way you choose (if you don't care about others using your extensions). XBRL is modular. Use what you see fit. Each feature adds a specific type of value, that is, it has a purpose. If your application of XBRL requires that feature, then use that marginal piece of functionality.

In this section we have endeavoured to explore and explain the various extensibility mechanisms available within XBRL. Understanding the different extensibility mechanisms is critical to selecting the correct mechanism to use for your specific circumstances. It also helps users see and realize the real values of XBRL.

While open standards may not provide total solutions which are superior to a specific proprietary solution, open standards to provide a base which can be leveraged which reduces costs of a total solution. XBRL, an open standard, may not meet 100 percent of any one organization's needs, but it certainly offers a robust, sound, global standard basis; a foundation which can be extended. Selecting the appropriate method of extensibility is driven by need. The more you need to share with others, the more important it is to use methods which make sharing possible – even easy.