2011

Navigating XML Documents

John W. Stamey
Coastal Carolina University, jwstamey@coastal.edu

Joseph S. Russell
Coastal Carolina University, jsrussel@coastal.edu

NAVIGATING XML DOCUMENTS

John W. Stamey
Coastal Carolina University
jwstamey@coastal.edu

Joseph S. Russell
Coastal Carolina University
jsrussel@coastal.edu

ABSTRACT

XML has become an important standard way for transporting interoperable data. As more applications require XML to import and export data, a greater understanding of XML and associated technologies will be required to work in this environment. This paper will discuss the parsing and navigation of multiple XML documents where data elements in instance documents are associated with data elements in XML linkbase documents.

Keywords

XML, XLink, XBRL, Linkbase, Python

INTRODUCTION

A number of standards for data sharing are relying on XML. RSS feeds are constantly used to share news stories. XBRL will be the standard for reporting and sharing accounting data for all publically traded firms by 2011. (SEC, 2001; Stamey and Casselman, 2009) As applications such as HL7 (Health Level Seven), FpML (Financial Products Markup Language), and XBRL (eXtensible Business Reporting Language) become more prevalent, more XML programming skills will be needed for the analysis and comparison of the data being collected. A recent example of the ease of comparing earnings reports between two major computing companies (Apple and Dell) using XBRL filings, was publicized by WebFilings. (Ritz, 2010)

A typical example of using XBRL data would be the analysis of earnings reports by a sell-side analyst, typically working for a brokerage company managing individual accounts such as Prudential or Fidelity. If a group of companies such as gold mining corporations report their earnings over a period of several days, analysts will take advantage of their information reported in XBRL to automate the comparison of earnings across this group of stocks. Such an automated comparison could take minutes whereas a comparison of the data in traditional text files reported to the SEC could take up to several weeks. An enormous financial advantage can be achieved for investors and analysts resulting from both reduced cost of information as well as the ability to perform immediate and timely analysis of financial information. (XBRL US, n.d.)

Functionality	Number	Percent
Taxonomy Creation/Extension	11	36.66%
Instance Creation and Validation	16	53.33%
Taxonomy Validation and Comparison	11	36.66%
Financial Information Analysis	7	23.33%
Advanced Reporting Analytics	2	6.66%
Data Management	6	20%
Instance Rendering and Printing	10	33.33%

Table 1. Availability of Functionality in XBRL Software

Data describing commercially available XBRL software (XBRL US, 2009) was anal;yzed, with the results reported in Table 1. It was found that of the thirty known and endorsed software products, only two of these packages had any sort of functionality to compare XBRL documents from different companies (Advanced Reporting Analytics, 6.66%). With currently only 37% of the data elements being directly (pair wise) comparable between two XBRL US-GAAP filings (Zhu and Wu, 2010), the time and cost savings goals of XBRL will most likely be realized through custom software. In general, custom software solutions will be written in a high-level programming language (such as Python or Java) and will require the use of class libraries for manipulation of a variety of XML-related technologies.

This paper begins with a discussion of XLink, and then proposes a workable design solution for parsing and searching XML linkbase documents such as those typically found in XBRL filings. The language chosen is Python, which has become a popular choice for XML programming due to its object-oriented nature and superior text manipulation capabilities. (Jones and Drake, 2002) XML methods used to extract data and attribute information include `getAttribute` and `getElementByTagName`. The `data` property is used to extract the information (formally called *XML data*) found between the opening and closing XML tags. Relationships between XML elements defined with XLink tags will also be discussed. We conclude with a practical example of parsing and navigating multiple XML files from a live XBRL filing.

PARSING XML DOCUMENTS IN PYTHON

According to www.Python.org, the official website of the Python Programming Language, there are three ways to parse XML in Python:

- SAX: unidirectional, event-driven parsing of XML documents node by node;
- DOM (and Minidom): converts an XML document into an in-memory tree object, allowing forward searching, and can be manipulated with method calls on nodes; and,
- ElementTree: a lightweight (and more "Pythonic") XML parser by Fredrik Lundh.

Our parsing approach will use Minidom, due to its similarity to the Document Object Model and our familiarity with the technology. Four important commands to be used in this example will:

- Import the Minidom library: `from xml.dom import minidom`
- Parse an XML document into an array : `dom = minidom.parse("document.xml")`
- Retrieve data from an element, given a namespace:
```
NAMESPACE= 'http://www.URIofNamespace.com'
x = dom.getElementsByTagNameNS(NAMESPACE, 'nameOfTag')
print x
```
- Retrieve attributes from tags, given a namespace:
```
NAMESPACE= 'http://www.URIofNamespace.com'
x = dom.getAttributeNS(NAMESPACE, 'nameOfAttribute')
print x
```

NAVIGATING BETWEEN DOCUMENTS WITH XLink

We refer to XML documents containing primarily data as *instance documents*. Elements in XML documents can be associated, or linked, to elements within the same XML document or to elements in a different XML document using XLink, a W3C specification. (W3C, 2010) Depending upon the architecture of the XML files, one would typically put the links between XML elements in a separate document called a *linkbase*. The XLink specification provides five types of attributes that provide a rich set of behaviors for links. (Arciniegas , 2000) These are found in Table 2.

Attribute Functionality	**Attribute Description**
Type	`simple` (standard link), `extended` (multi-resource), `locator` (point to external resource), `resource` (point to internal resource), `arc` (traversal rule between resources), `title`
Locator	`href` (location of a resource)
Semantic	`role` (how used),
Behavioral	`show` (where/what window to open), `actuate` (when to open)
Traversal	`label` (name), `from` (beginning of arc), `to` (ending of arc)

Table 2. XLink Attributes and Functionality

A traditional XHTML hyperlink would be coded as follows: (Stamey and Russell, 2010)

```
<a href="http://www.XML.com">Information about XML</a>
```

The same idea, implemented using XLink, would be:

```
01 - <?xml version="1.0"?>
02 - <XBRLresource
03 -    xmlns:XLink="http://www.w3.org/1999/XLink/"
04 -    xlink:type="simple"
05 -    xlink:href="http://www.xbrl.com"
06 -    xlink:show="new"
07 -    xlink:actuate="onRequest">Information about XML</anchor>
```

The additional attributes contribute to the semantics of this element as follows:

Line	Description
03	Specifies the xlink namespace
04	Defines the link to be of type simple (a traditional hyperlink)
05	Gives the URI to which we will navigate
06	Requires the URI will open in a new window
07	States the new windown will opwn when the text is clicked upon

Table 3. Description of Attributes in a simple XLink

A PRACTICAL EXAMPLE

As an example of XLink, we have chosen an XBRL filing from NYSE:AEO, American Eagle Outfitters, Inc., (SEC, 2010) which contains an instance document along with several linkbases. We will examine an element in Figure 1, from the instance document `aeo-20101030.xml`.

```
<us-gaap:TreasuryStockValue contextRef="BalanceAsOf_31Oct2009"
                unitRef="USD" decimals="-3">760197000</us-gaap:TreasuryStockValue>
```

Figure 1. An XML data element from the AEO instance document

For this example, we are interested in tags from the label linkbase, which provide nicely formatted text to accompany the data in the instance document. The associated XLink tags from the label linkbase, `aeo-20101030_lab.xml`, are seen in Figure 2.

```
01 -  <loc xlink:type="locator"
02 -  xlink:href="http://taxonomies.xbrl.us/us-gaap/2009/elts/
03 -          us-gaap-2009-01-31.xsd#us-gaap_TreasuryStockValue"
04 -  xlink:label="us-gaap_TreasuryStockValue" />

05 -  <labelArc xlink:type="arc"
06 -  xlink:arcrole="http://www.xbrl.org/2003/arcrole/concept-label"
07 -  xlink:from="us-gaap_TreasuryStockValue"
08 -  xlink:to="lab_TreasuryStockValue" />

09 -  <label xlink:type="resource"
10 -  xlink:role="http://xbrl.us/us-gaap/role/label/negatedTotal"
11 -  xlink:label="lab_TreasuryStockValue"
12 -  xml:lang="en-US">Treasury Stock, Value</label>
```

Figure 2. A collection of related XLink tags from the AEO linkbase

The tag `us_gaap:TreasuryStockValue` from the instance document (Figure 1) contains the numerical data, while the tags in the linkbase (Figure 2) guide us to the English-readable description of this data (`Treasury Stock, Value`). The linking is straightforward, viewing Figure 2, from line 04 → line 07, then from line 11 → line 12. We now examine this code in a more formal manner.

The `loc` element from the label linkbase document.

In the label linkbase (lines shown in Figure 2), the `loc` element has the following three attributes:

- `type="locator"` which means the tag points to an external resource
- `href="URI#us-gaap_TreasuryStockValue"` specifies an internal link in a URI which has some information
- `label="us-gaap_TreasuryStockValue"` is a traversal attribute and gives us the name of a tag and its namespace (here, the namespace is `us-gaap` and the tag is `TreasuryStockValue`).

This tag would be described as follows: "We define a value us-gaap_TreasuryStockValue. Information about this value can also be obtained at http://URI#us-gaap_TreasuryStockValue/".

The `labelArc` Element From The Label Linkbase Document

A `labelArc` element contains the relationship between `us-gaap_TreasuryStockValue` (the element in the instance document) and `lab_TreasuryStockValue` (the element in the label linkbase with the human-readable formatted text). In particular, the attributes of the `labelArc` tag are:

- `type="arc"` is a traversal rule between resources.
- `arcrole="A URI"` which references a description of the arc role.
- Attributes from and to, which indicate the start and end points of the arc.

This tag would be described as follows: "The TreasuryStockValue element from the US-GAAP namespace in the instance document is related to the lab_TreasuryStockValue label in the label linkbase document."

The `label` element from the label linkbase document

The `label` element associates a label with human-readable text. Attributes of the label tag are:

- `type="arc"` means this tag contains a traversal rule (relationship) between resources.
- `role="URI"` is a semantic attribute stating the location of documentation.
- `label="lab_TreasuryStockValue"` specifies the name of the label.
- `lang="en-US"` identifies English as the language for the data in this element.

Figure 3 graphically depicts the three relationships the label linkbase and the data element in the instance document.

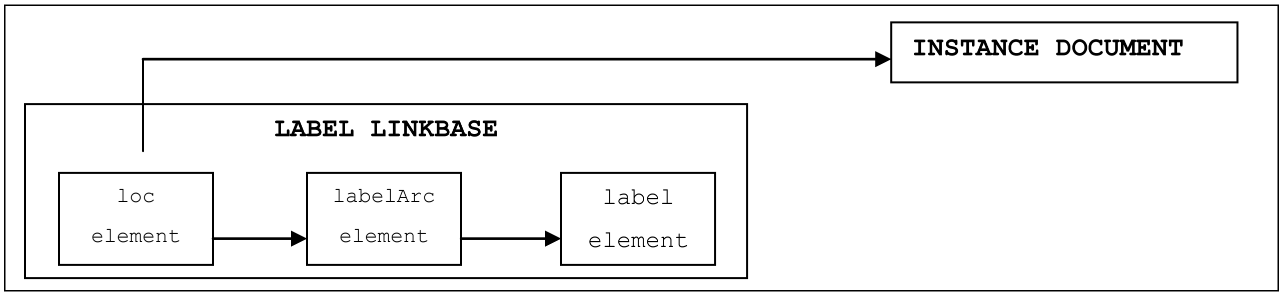

Figure 3. Relation of XLink tags in Label Linkbase to Instance Document

PROGRAMMING PROCEDURE AND IMPLEMENTATION

Pseudocode describing the procedure outlined above is found in Figure 4. The Python 3.1 code to implement the pseudocode is found in Figure 5. Recognizing that we actually have the attribute value of `to` (the from `loc` element) in the `labelArc` element, we begin our parsing and navigation from that point.

```
set namespaces for XLink and US-GAAP (lines 02-03)
parse the label linkbase with Minidom (line 05)
parse the instance document with Minidom (line 06)
ITERATE through loc tags in the label linkbase
   IF(attribute "label" in labelArc tag == attribute "from" in labelArc tag
      IF(attribute "to" in labelArc tag == attribute "label" in label tag)
         find this tag in the instance document
         format data from instance document based on its tag attributes
         print formatted data and data value of label tag
```

Figure 4. Pseudocode for matching instance data with English-readable value in label linkbase

```
01 - from xml.dom import minidom
02 - XLINK_NS='http://www.w3.org/1999/xlink'
03 - USGAAP_NS='http://xbrl.us/us-gaap/2009-01-31'
04 - def main():
05 -     y = minidom.parse("../AmericanEagle/aeo-20101030_lab.xml") # parse label linkbase
06 -     z = minidom.parse("../AmericanEagle/aeo-20101030.xml") # parse instance document
07 -     for node in y.getElementsByTagName('labelArc'):
08 -         fromAttribute = node.getAttributeNS(XLINK_NS, 'from')
09 -         toAttribute = node.getAttributeNS(XLINK_NS, 'to')
10 -         humanReadable = ""
11 -         for node1 in y.getElementsByTagName('label'):
12 -             if (node1.getAttributeNS(XLINK_NS, 'label')==toAttribute):
13 -                 humanReadable = node1.firstChild.data
14 -         outputString = "INFO - " + humanReadable
15 -         print (outputString)
16 -         isItText = fromAttribute.find("TextBlock") # TextBlock implies all HTML comments
17 -         if (isItText < 0):   # If TextBlock is not found (means we have data to report)
18 -             if(fromAttribute[0:8]=="us-gaap_"):
19 -                 theLength=len(fromAttribute)
20 -                 theTag = fromAttribute[8:theLength]
21 -             for node2 in z.getElementsByTagNameNS(USGAAP_NS, theTag):
22 -                 try: # Do this in case there is no data
23 -                     theAmount = node2.firstChild.data
24 -                 except: # Required: If there was no data, say so!
25 -                     theAmount = "No Data"
26 -                 theContext = node2.getAttribute('contextRef')
27 -                 theDecimals = node2.getAttribute('decimals')
28 -                 theUnitRef = node2.getAttribute('unitRef')
29 -                 theOutput = "\"" + humanReadable + "/"" + "Amount: " + theAmount +
30 -                        " Decimals:" + theDecimals + " Context:" + theContext +
31 -                        " Units:" + theUnitRef
32 -                 print(theOutput)
33 -         print("  ")
34 - main()
```

Figure 5. Python Code to Parse and Traverse with XLink

An application of this algorithm would result in the following formatted output:

```
Treasury Stock, Value         -760,197  USD
```

CONCLUSION

In this paper, we have shown a valuable technique for associating data elements in an XML instance document with data elements in an XML linkbase document. The once novel use of XLink in the XBRL standard, is now on its way to becoming a standard way of linking data elements between different XML files. (DuCharm3, 2002; vun Kannon, 2007) The increased use of XML for storing and sharing data will require programmers to develop more skills, such as those we have shown, in this paper.

REFERENCES

1. Arciniegas, F. (2000) Xlink Reference. Retrieved November 1, 2010 from http://www.xml.com/pub/a/2000/09/xlink/part2.html/.
2. DuCharme, R. XLink: Who Cares? Retrieved January 4, 2011 from http://www.xml.com/ pub/a/2002/03/13/xlink.html/.
3. Jones, C.A. & Drake, F.L. (2002) Python & XML. O'Reilly Media.
4. Mertz, D. (2001) Understanding ebXML. Retrieved January 2, 2011 from http:// www.ibm.com/developerworks/xml/library/x-ebxml/.
5. Ritz, D. (September 30, 2010) Comparing Year 2 Filers, Apple and Dell, using the WebFilings XBRL Taxonomy Analyzer. Retrieved January 9, 2011 from http://www.webfilings.com/blog/comparing-year-2-filers-apple-and-dell-using-webfilings-xbrl-taxonomy-analyzer/.
6. Securities and Exchange and Commission (February 10, 2001) Interactive Data to Improve Financial Reporting: Final Rule (17 CFR Parts 229, 230, et. al.) Federal Register. Retrieved October 1, 2010 from http://www.sec.gov/rules/final/2009/33-9002fr.pdf/.
7. Securities and Exchange Commission (2010) American Eagle Outfitters, Inc. 10-K Filing, Retrieved November 19, 2010 from http://www.sec.gov/Archives/edgar/data/919012/000095012310107439/0000950123-10-107439-index.htm/.
8. Stamey, J. and Casselman, C. (2010) Reporting and Sharing Financial Data with XBRL, *Proceedings of Southern Association of Information Systems Conference*, Atlanta, GA.
9. Stamey, J. and Russell, J. (2010) Introduction to XBRL, Digital University Press, Conway, SC.
10. W3C: World Wide Web Consortium (2010) XML Linking Language (XLink) Version 1.1. Retrieved December 1, 2010 from http://www.w3.org/TR/xlink11/.
11. XBRL.US Laboratories (n.d.) Member Products and Services. Retrieved January 9, 2011 from http://XBRL.us/learn/pages/gatalog.aspx.
12. XBRL US Laboratories (n.d.) Improving the Analytical Process with XBRL. .Retrieved January 9, 2011 from http://xbrl.us/Learn/Documents/XBRLforAnalysts.pdf/.
13. vun Kannon, D. (2004) XLink Alive and Wekk in XBRL. Retrieved January 4, 2011 from http://www.xml.com/ cs/user/view/cs_msg/1828/.
14. Zhu, H. and Wu, H. (March 25, 2010) XBRL and Interoperability of Financial Statements in the U.S. Available at SSRN: http://ssrn.com/abstract=1581511/.