The International Journal of Digital Accounting Research
Vol. 2, No. 1, pp. 83-118
ISSN: 1577-8517

On the design of an XML-Schema based application for business reporting: An XBRL Schema Perspective[1]

Kinsun Tam. State University of New York at Albany.

Sanjay Goel. State University of New York at Albany.

Jagdish S. Gangolly. State University of New York at Albany

Abstract. A markup language for business reporting must satisfy many demanding criteria: readable by novices, extendable by users, minimum payload overheads, and a uniform graph structure to enable validation of document instance with minimal programming effort. To be elegant and robust it must be based on a model that reflects the intricacies of business reporting, and to be efficient in terms of maintenance it must be modular in structure. We suggest the skeleton of a derivative of the XBRL that exhibits most of the criteria stated above which uses the basic semantic structure provided in its specification and the associated C&I taxonomy. Our proposal provides domain-specific tags so that even the source documents are very readable. We provide a proof-of-concept schema for the Balance Sheet (using the XBRL C&I taxonomy) as an instance of a canonical generic labeled graph model for any financial statement. We also provide an algorithm for the validation of such labeled directed graph representation of a financial statement and its implementation in the programming language Java.

Key words: *Markup languages, Business reporting language, Validation of financial reports*

[1] The work of K. Tam was supported by a Faculty Research Award Program Grant of the State University of New York at Albany. J. Gangolly gratefully acknowledges the insights into XBRL architecture provided by Mr. David vun Kannon (the Chief architect of XBRL) of KPMG Peat Marwick in private communications as well as on the XBRL-public group exchanges, and to Professor Neal Hannon of Bryant College for facilitating the exchanges. Mr. Kannon's comments on an earlier version of the paper are also gratefully acknowledged.

1 INTRODUCTION

The globalization of the capital markets, the fast growth of the Internet (particularly its world-wide web component), the evolution of B2B commerce, and the momentum gained by the protocols based on open standards have set the stage for the development of XML based languages to support business exchanges of data and business reporting (W3C, 2000). One specific language of interest to us in accounting is the Extensible Business Reporting Language (XBRL, vun Kannon & Wang, 2001) initiative of the AICPA. Over the past two years, XBRL has gained considerable international exposure and support from various stakeholders in the business reporting process.

The importance of protracted deliberations and serious reflection in the development of markup languages for business reporting becomes apparent when one considers the enormity of the impact of online delivery of business reports on the Internet infrastructure. For example, in the United States alone, there are over 10,000 corporations listed on one of the stock exchanges, and each is required to make numerous filings under the securities laws alone. While at present the use of such data online by individual naive investors is perhaps not overwhelming, the development of applications (integrating business reports) facilitated by XBRL should result in considerable growth in the online use of XBRL compliant data by naïve investors.

Our goals in this paper are to state and justify certain sound principles of XML-Schema (W3C, 2001a, 2001b, and 2001c) based application design, and to illustrate their use in the context of one financial statement: the Balance Sheet. We construct a schema that is modular, based on the definition and use of type hierarchy specific to the Balance Sheet, but invariant with respect to other financial statements in terms of its overall structure. This proposed schema provides domain-specific tags, is extensible to individual taxonomies, and facilitates a standard algorithm for the validation of schema instances of all financial statements. It also minimizes “payload overhead”, i.e, the percentage of data in a document that is not payload.

Our objective here is not to propose yet another tagging scheme, but to provide the sketch of a markup language that in our opinion is elegant and imposes minimal

burden on the preparer of the reports, the readers of the document source, as well as the Internet infrastructure. We provide enough evidence, by way of an XML-Schema compliant schema, and an algorithm for the validation of instance documents complying with our proposal.

In addition, our objective here is not to put forth a comprehensive proposal for a new language. In fact, our schema can be used in conjunction with the XBRL DTD. It is to suggest a way in which the value of the XBRL language can be enhanced. Therefore, we do not consider meta-level issues such as the specification of the context (part of which is in the parameter entity att_AttributeHolder in the XBRL specifications), labels associated with elements, and statements with multi-year data. Those matters await a sequel, in the works, to this paper. We also do not address the very important issue of co-existence with other initiatives such as XFI (XML Fragment Interchange, W3C Consortium, 2001), DAML (DARPA Agent Markup Language, DAML, 2000), the Semantic Web initiative (Berners-Lee et al., 2001), CKML (Conceptual Knowledge Markup Language, Kent, 2001), RDF (Resource Description Format, W3C, 2002), and the like. These matters too are in the works as sequels to this paper. These outstanding issues, in our opinion, do not fundamentally alter the overall design presented here.

In Section 2 we state the requirements that we considered in the design of our schema. While the XBRL architects considered requirements for a business reporting language at a sufficiently high level, we consider specific requirements at the software architectural level. In Section 3, we describe the graph model for business reporting and its realization in the schema for Balance Sheet. In Section 4 we provide the algorithm for the validation of business reporting instance documents and its implementation in java. Finally, we provide concluding observations in Section 5. The appendices contain the UML (class diagram) model, the schema definition, a schema extension to show its extensibility, a basic XSL style sheet, an instance document, and the validation routine in java.

2 REQUIREMENTS FOR A MARKUP LANGUAGE FOR BUSINESS REPORTING

An XML-Schema is the blueprint for an XML document in the same way that a relational schema is the blueprint for a relational database. The Design of XBRL Specification provides examples of business, political, as well as technical requirements. In this paper, however, we concentrate on the technical requirements. We will very briefly state the requirements underlying the Design of XBRL Specification paper, and then provide detailed requirements that, in our opinion, are important for its long-term success.

The Design of XBRL Specification states three basic technical requirements: extensibility through reuse via incremental extensions, format consonant with securities filing requirements, and substance over form (in the sense of lack of text formatting requirements in the specifications). All three are quite important from the point of view of the preparers and users of the reports. However, in our opinion, there are at least six additional requirements to conform to good software engineering practices. We will briefly discuss them below:

Based on standard modeling methods: In so far as the XML-Schema is the blueprint for an XML instance document, it is important that it be based on a rigorous model. While one can model the schema using techniques such as conceptual graphs, concept diagrams, ODL (Object Definition Language), we chose UML class diagrams since they are perhaps sufficiently expressive in meeting our modeling needs, and that they are most widely used modeling language in software development. In fact, UML models were used in the XBRL design, but at a very high level, to yield three labels: item, group and label. Presently, XML does not incorporate object-oriented features[2] (inheritance, information hiding, etc.) and therefore our UML object model does not add value to the implementation. However, when object-oriented XML does materialize, we will be able to take full advantage of UML object modeling.

[2] However, there is a proposal before the WWW Consortium on Schema for Object-oriented XML (SOX). See *Schema for Object-Oriented XML 2.0* (http://www.w3.org/TR/NOTE-SOX/, visited on May 15, 2002).

Common schema structure for all financial statements so that validation routine can be standardized: An important supposition of the XBRL design is that the validation of an instance document is an application layer concern. While this is quite true, the choice of a graph model for schema has important consequences for all applications. It is therefore important to choose models and data structures that result in efficient development of validation software. If there is a common graph model for the schemas for the various financial statements, the development of a single comprehensive validation algorithm is facilitated. Instance validation based on domain-specific rules at the browser level is important to provide users comfort and also build trust, especially since the data in XBRL is largely of a financial nature.

Readability of XML code: The readability of source document is also improved when it contains just the minimum information to be displayed; all other information is hard-wired in the schema. Moreover, as will become clear in the subsequent sections, the readability of the schema is also improved by typing of the nodes. Typing refers to separating out verbose data type definitions so that the remaining tree structure becomes more readable.

Extensibility of schema: This is substantially the same as the requirement in the XBRL Design document. Since it is unrealistic to have a schema satisfying all user requirements, the schema should be extensible by individual users.

Modular schema so it is easier to maintain: The schema should be modular so that any changes are easily incorporated. Typing of the nodes facilitates it.

Minimization of payload overheads: The information in an XBRL instance consists of the information to be displayed as well as structural information. A simple structure can reduce the payload overheads for structural information.

In the next section, we provide the canonical design for a financial report and its implementation for balance sheet. We will refer to the requirements in our discussion of the design.

3 THE DESIGN

Financial reports are perhaps the primary means of communicating financial information to the public. While their use at the present time by naïve investors may not be widespread, as the Internet technologies become ubiquitous and applications using such online information become prevalent, one can expect their use to be pervasive. Therefore, it is important that there be a canonical design of data structures (graph models) for financial reports to facilitate the development of algorithms for their validation.

In this section, we provide a canonical graph model for financial reports, and discuss the UML class diagram for the balance sheet and the associated XML-Schema. The next section will argue that this model provides a standard validation algorithm for all financial statements.

Most financial reports have a similar hierarchical structure that lends itself to a canonical graph (Directed tree) model. Since XML-Schema as well as XML instance documents are also hierarchical in structure, modeling the schema for financial reports as a canonical tree permits us to exploit the DOMTree generated by parsing an XML instance to extract the structure in order to perform the validation of financial reports.

We represent all financial reports (with the possible exception of footnotes, which we have yet to model) as a tree shown in Figure 1 below. Each node in the tree represents either an element (say, a balance sheet item for example) or an attribute. We distinguish between them by writing the labels of element nodes in upper case letters and the labels of attributes in lower case letters. Each element node has three attributes: parent, relation, and balance.

The parent attribute specifies the name of the parent element node. The set of element nodes and the parent attribute for each such node specifies a financial statement directed tree.

The balance attribute specifies the balance amount of the element node. The value of the balance attribute for each element node is displayed in the financial statement on the browser.

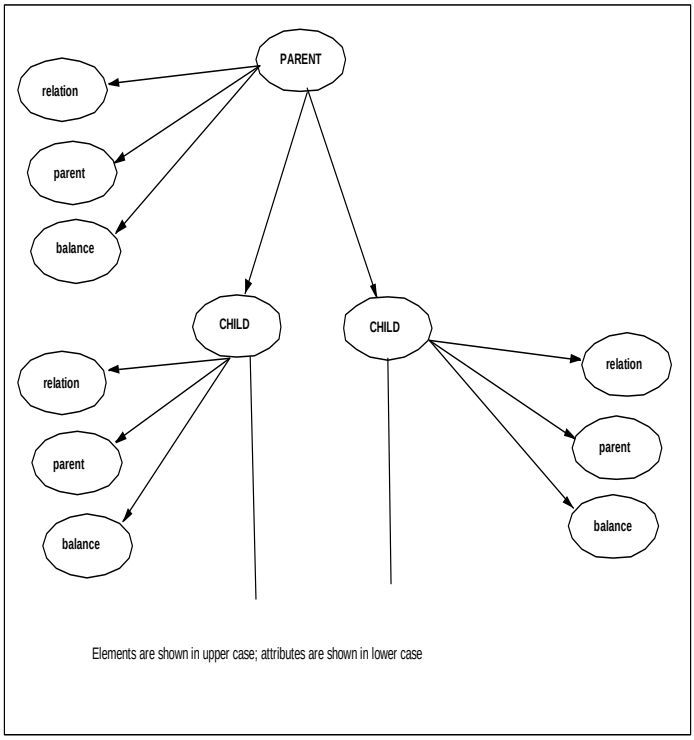
The relation attribute of an element node specifies the relationship of the element balance to the balance amount of the parent element node. If the value of relation is 0, then the balance attribute of an element node must be equal to the balance attribute of the parent element node. On the other hand, if it is –1, then the contribution of the balance attribute of an element node towards the balance attribute of the parent element node is negative. Analogously, if the relation attribute is +1, then the contribution of the balance attribute of an element node towards the balance attribute of the parent element node is positive. Because treasury stock is to be deducted from stockholders' equity, for example, the relation attribute of the treasuryStock node is -1, indicating a negative contribute to its parent node.

For a financial statement directed tree to be valid, for each element node in the tree, the value of the balance attribute must equal the sum of its children node balance attribute adjusted for sign (using the children nodes relation attribute values). For instance, the sum of the balance attributes of the currentAssets and nonCurrentAssets nodes must equal the balance attribute of the totalAssets node. The parent and relation attribute essentially perform the same role as the rollup and weight attributes in the XBRL specification.

The model in Figure 1 provides a template for generating XML-Schema for financial reports from the UML class diagrams (we used Rational Rose Enterprise Edition version 7.1). This is what we discuss next.

Figure1

The Canonical graph (Directed Tree) model for Financial Statements



Appendix 1 provides the UML class diagram for a Balance Sheet. We used the terminology given in the XBRL C&I taxonomy, but stopped at level 5 in most cases since our objective was to have a proof-of-concept for our model. Figure 2 below provides the associated graph model. Appendix 2 gives the schema (we used XML Spy version 3.5) based on the UML class model and the graph model.

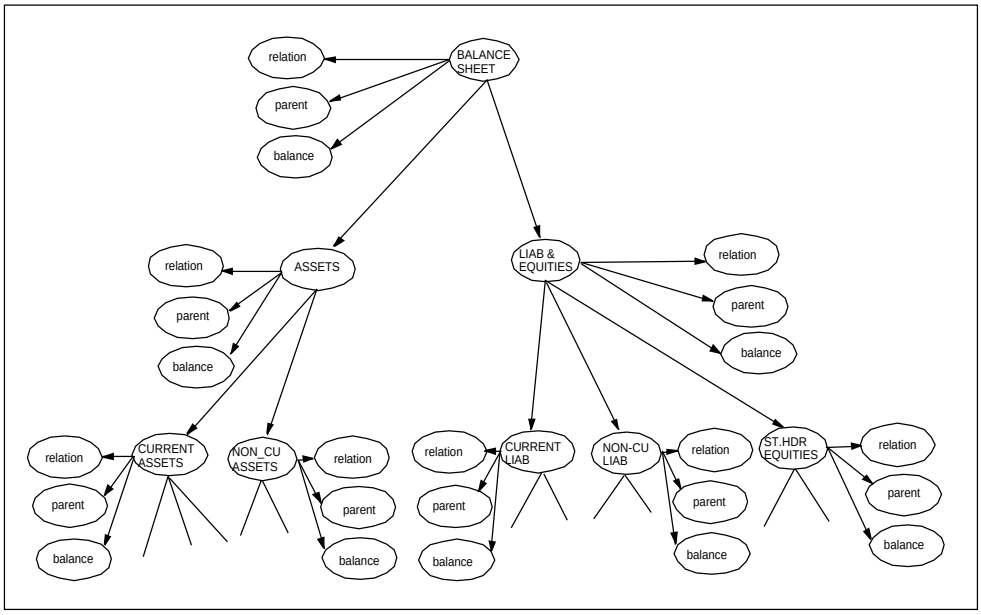


FIGURE 2. The Canonical Graph (Directyed Tree) Model for Balance Sheet

The schema in Appendix 2 is based on the definitions of type for each node in the hierarchy. At the top level the element balanceSheet is defined to be of the balanceSheetType, which is defined to consist of the three attributes of the balance sheet (relation, parent, and balance) and two sub-elements (totalAssets and liabilitiesAndEquities) as follows:

```
   <!-- Definition of the root element balanceSheet-->
   <element name="balanceSheet" type="xfs:balanceSheetType"/>
   <!-- Definition of the balanceSheetType-->
   <complexType name="balanceSheetType">
      <all>
         <element name="totalAssets" type="xfs:totalAssetsType"
minOccurs="0"/>
         <element name="liabilitiesAndEquities"
              type="xfs:liabilitiesAndEquitiesType"
minOccurs="0"/>
      </all>
      <attribute name="relation" type="integer" fixed="0"/>
      <attribute name="balance" type="decimal" use="required"/>
      <attribute name="parent" type="NMTOKEN" fixed="statement"/>
   </complexType>
```

Content models for these sub-elements are in turn defined in terms of the sub-elements and the three attributes. For example, the element totalAssets and liabilitiesAndEquities are defined in terms of its child elements as follows:

```
<!-- Definition of the totalAssetsType-->
<complexType name="totalAssetsType">
   <all>
      <element name="currentAssets" type="xfs:currentAssetsType"
minOccurs="0"/>
      <element name="nonCurrentAssets"
type="xfs:nonCurrentAssetsType"
            minOccurs="0"/>
   </all>
   <attribute name="relation" type="integer" fixed="0"/>
   <attribute name="balance" type="decimal" use="required"/>
   <attribute name="parent" type="NMTOKEN" fixed="balanceSheet"/>
</complexType>

<!-- Definition of the liabilitiesAndEquitiesType-->
<complexType name="liabilitiesAndEquitiesType">
   <all>
      <element name="currentLiabilities"
            type="xfs:currentLiabilitiesType" minOccurs="0"/>
      <element name="nonCurrentLiabilities"
            type="xfs:nonCurrentLiabilitiesType" minOccurs="0"/>
      <element name="stockholdersEquity"
            type="xfs:stockholdersEquityType" minOccurs="0"/>
   </all>
   <attribute name="relation" type="integer" fixed="0"/>
   <attribute name="balance" type="decimal" use="required"/>
   <attribute name="parent" type="NMTOKEN" fixed="balanceSheet"/>
</complexType>
```

Ideally, the attributes balance and relation should be globally defined as in the box below. However, Xerces 1.4 does not implement the infoset contribution (supplementary information that is not immediately present in the instance, such as types and default values) correctly. Only attributes with unqualified (un-prefixed) names are added as infoset contribution. Accordingly, though an inelegant solution, we have had to use simple types for relation and balance. This changes relation and balance to non-root level attributes, for which the xfs prefix is not needed in the instance documents. This is reflected throughout our schema and our discussions.

```
<!-- "balance" is the amount as would appear on the Balance
Sheet.-->
<attribute name="balance" id="balance" type="decimal"/>

<!-- Relation is the nature of the relationship with the
    "parent" element.

    relation =  0 : the "balance" equals the balance of the
                      "parent" element

    relation = +1 : the contribution of the element "balance"
                    to the balance of the "parent" element
                    is positive
    relation = -1 : the contribution of the element "balance"
                   to the balance of the "parent" element
                    is negative  -->

<attribute name="relation" id="relation">
    <simpleType>
        <restriction base="integer">
            <minInclusive value="-1"/>
            <maxInclusive value="1"/>
        </restriction>
    </simpleType>
</attribute>
```

The values of the relation attribute are hardwired into the schema, though not globally, as can be seen from our schema, and the values of the balance attribute are bound in the instance document as can be seen in the example document in Appendix 5.

Schema extensibility is achieved through redefining types. The schema in Appendix 2, based on the UML model in Appendix 1, specifies a five level deep balance sheet hierarchy. Individual users who need to use accounts below the fifth level, such as child accounts of the netPropertyPlantEquipment node, can extend this schema to describe lower level nodes. The original specification of the type netPropertyAndEquipmentType, as provided in the balance sheet schema in Appendix 2, is reproduced as follows:

```
<!-- single out this netPropertyPlantEquipmentType for subsequent
redefinition in imb4.xsd-->
<complexType name="netPropertyPlantEquipmentType">
   <attribute name="relation" type="integer" fixed="+1"/>
   <attribute name="balance" type="decimal" use="required"/>
   <attribute name="parent" type="NMTOKEN"
fixed="nonCurrentAssets"/>
</complexType>
```

However, the schema in Appendix 2 specifies only the content model for this type in terms of the attributes relation, parent, and balance but not the child nodes. Child nodes corresponding to elements in the XBRL hierarchy below level 5 are added, through using the <redefine> element provided by the XML-Schema specification, in the schema extension in Appendix 3. For example, child nodes such as grossPropertyPlantEquipment and accumulatedDepreciationAndAmortization are specified through redefining as below:

```
<redefine schemaLocation="bsj6.xsd">
   <complexType name="netPropertyPlantEquipmentType">
      <complexContent>
         <extension base="xfs:netPropertyPlantEquipmentType">
            <all>
            <!-- Definition of the grossPropertyPlantEquipment-->
      <element name="grossPropertyPlantEquipment" minOccurs="0">
      <complexType>
            <all>
      <element name="land" minOccurs="0">
      <complexType>
       <attribute name="relation" type="integer" fixed="+1"/>
        <attribute name="balance" type="decimal" use="required"/>
      <attribute name="parent" type="NMTOKEN"
fixed="grossPropertyPlantEquipment"/>
      </complexType>
      </element>
                        .....
                    </all>

      </complexType>
</element>
<element name="accumulatedDepreciationAndAmortization"
                    minOccurs="0">
         <complexType>
   <attribute name="relation" type="integer" fixed="-1"/>
   <attribute name="balance" type="decimal" use="required"/>
   <attribute name="parent" type="NMTOKEN"
   fixed="netPropertyPlantEquipment"/>
      </complexType>
      </element>
```

```
                .....
            </all>
          </extension>
        </complexContent>
    </complexType>
</redefine>
```

When used in conjunction with the XSL stylesheet (Appendix 4) and the schemas (Appendices 2 and 3), the balance sheet instance document can be viewed in any XML supporting browser (we tested with the Internet Explorer version 5).

The design we have described above was based on an UML class model, based on a canonical directed tree model for all financial reports, and provides a tagset meaningful to accountants (having been based on the items in the XBRL C&I taxonomy). Since we have exploited the hierarchical nature of XML as well as the financial reports, the canonical representation enables us to write one validation routine for most financial reports. We have demonstrated the extensibility of schemas, and the readability of the code is obvious from the description above and in the appendices. The better readability is due to the modularization of the schema accomplished by defining types for the nodes in the report hierarchy.

In the next section, we describe the validation algorithm and its implementation.

4 VALIDATION OF FINANCIAL REPORTS: THE BALANCE SHEET

There are five components to the validation of instance documents:

1 Validation of the Schema: The main schema would be used by all preparers of financial statements. Since schema is written to comply with XML-Schema specifications, validation, if provided by browsers, should be adequate.

2 Validation of schema extensions: These would include lower level details in the financial statements for which schema extensions would be developed by the preparers of the financial reports. Since these too are written to comply with XML-Schema, validation, if provided by browsers, should be adequate,

3 Validation of stylesheets: The stylesheets for the presentation of the reports are written either complying with XSL or CSS specifications. The browser validation, if provided, here too can be relied upon.

4 Validation of financial report instance against schemas and schema extensions: Since they are written in XML, the browser validation, if provided, here too can be relied upon.

5 Validation of the financial reports: Since there is an internal structure to each financial report, it is important to verify that the rules governing such internal structure are not violated by a financial report instance. Such rules are dictated by accounting & auditing considerations and include rules for footing as well as those for consistency of articulated statements.

While the first four components above are generic to all XML-Schema/XSL/CSS based applications, the financial report validation component is specific to such reports. Consequently the usual xml validation facilities are provided by the browser. Since the semantics of such financial report validation rules are to be separately programmed, from the efficiency standpoint it is important that they be considered in the design of the schema. As the use of financial reports gets embedded in user applications, such validation becomes crucial to provide assurance to the naïve investors.

In as much as the financial reports contain information that is articulated in certain ways, their integrity depends crucially on the compliance with the rules of such articulation. For example, since the two sides of a balance sheet must balance, it is important that total assets equal total liabilities & equities. Similarly, the total assets must equal the sum of all the individual assets, and so on. The credibility of a balance sheet, like that of all financial reports, hinges crucially on whether it complies with these rules. Ultimately, if reliance is placed by the users of financial statements while browsing them on the Internet, it is imperative that there be browser level validation of financial reports (for compliance with such rules of articulation) each time they are to be displayed. This can be implemented either by a browser-side plug-in, or via validation and certification at the server-end. In either case, it is important that the validation be done by a common program for all

financial reports in an efficient manner. In this section, we discuss the validation algorithm and its implementation in a java program.

The schema as well as the instance documents developed in this study, except for schema extensions and instance documents based on such extensions[3], were validated with respect to the W3C recommendations on XML and XML-Schema using XML Spy 3.5 and the Schema Quality Checker developed by IBM. We will concentrate here on the validation of the instance documents with respect to the articulation rules for financial statements.

Since any financial report can be cast into the canonical tree representation in our schema defined in the previous section, we can derive a simple tree-traversal algorithm for validation, which can be explained as follows:

> *Starting with the root node of the document,* **for each node** *in the DOMTree,* ***get*** *a list of child nodes and their attribute names & values.* ***If*** *the value of the child node relation attribute value is "0", the value of the balance attributes for such node and the child must be equal,* ***else*** *the sum of the children node balance attribute values weighted by the value of the relation attribute must be equal to the value of the balance attribute of the node.*[4]

The implementation of this algorithm consists of five java classes: FinancialStatement, Util, Parse, DOMErrorHandler, and Validate. The FinancialStatement class accepts command line input for the report (XML instance document) to be validated, sets up the document for parsing and validating. It also has methods for getting the root node of the document, and printing the DOMtree for the document.

The Util class contains the methods for printing the tree and the nodes, converting the filenames to URIs (needed by the parser). The Parse class contains the methods to read the instance document into the Document Object Model and use the Xerces DOMParser. The Validate class contains all the methods used in

[3] Validation of extensions and instance documents based on schema extensions are not yet supported by XML Spy 3.5. However, Xerces parser does support these features. Therefore, they were validated through the Xerces parser.

validation, including that for recursive checking of the (footing) rules for financial reports. Given a node, while there are child nodes, the method checkCascadingBalance gets the NodeList of child nodes, and then recursively validates the financial statement as follows: if the relation attribute of the child node is 0 checks for compliance with the equality of node balance and the child's balance attributes by invoking the checkEqualityRelation method, otherwise gets the sum of the children nodes balance weighted by the value of their relation attribute by invoking the getChildNodesBalanceTotal method and checks the equality of the node balance and the weighted sum of child node balances.

```
/** Checks to see that for the entire tree the balance of the
parent is equal to
    the sum of balances of the children nodes. And recursively
checks this for
    each child node and so on.
    @Node node - Root node for tree which needs to be validated
    @return double - Value of the sum of balance attribute of
children nodes */

public static boolean checkCascadingBalance(Node node) {
    boolean isValid = true;
    System.out.println("checkCascadingBalance LEVEL " + ++index);
    System.out.println("Node Name " + node.getNodeName());
    // If node is child node just return;
    if (!node.hasChildNodes()) {
        System.out.println("Reached Leaf Node");
    }
    else {
        NodeList childList = node.getChildNodes();
        int length = childList.getLength();
        System.out.println("Number of Children = " + length);
        System.out.println("Parent Node Balance = " +
getNodeBalance(node));
        System.out.println("Printing child node balance");
        System.out.println("Children node balance = " +
                getChildNodesBalanceTotal(node));
        // If node has children get the sum of children and
compare
        System.out.println (" Difference = " +
                (java.lang.StrictMath.abs(getNodeBalance(node) -
                getChildNodesBalanceTotal(node))));
        if (checkIfEqualityRelationExists(node)) {
        isValid = checkEqualityRelation(node);
        System.out.println("Boolean from checkEqualityRelation = "
```

[4] A DOMTree is a tree object under the Document Object Model (DOM). DOM provides a set of application programming interfaces (API) for navigating a tree.

```
+
                        isValid);
        }
        else {
        if (java.lang.StrictMath.abs(getNodeBalance(node) -
                    getChildNodesBalanceTotal(node))<
allowedDifference) {
            // Set the boolean flag
            isValid = true;
        }
        else {
            isValid = false;
        }
    }
    if (isValid) {
    // Call recursively the same function for all children nodes.
        for (int i = 0; i < length; i++) {
            if (childList.item(i).getNodeType() == 1) {
                isValid =
checkCascadingBalance(childList.item(i));
                if (!isValid)
                    break;
                }
            }
        }
    }
    System.out.println("before returning isValid  = " + isValid);
    return isValid;
}
```

The method checkCascadingBalance in Validate.java which implements the validation.

The checkCascadingBalance method of the Validate class calls the method getChildNodesBalanceTotal method given in the box below. It gets the list of child nodes, for each child the values of the balance and relation attributes, and finally computes the weighted sum of the child balances.

```
public static double getChildNodesBalanceTotal(Node node)
{
    NodeList childList = node.getChildNodes();
    int length = childList.getLength();
    double sum = 0;
    for (int i = 0; i < length; i++) {
        if (childList.item(i).getNodeType() == 1) {
            Double dbal =
getNodeDoubleAttributeValue(childList.item(i),
                    "balance");
```

```
            if (dbal == null) {
                System.out.println("balance attribute not
speficied");
                System.exit(-1);
            }
            double balance = dbal.doubleValue();
            System.out.println("balance = " + balance);
            Double drel =
getNodeDoubleAttributeValue(childList.item(i),
                    "relation");
            if (drel == null) {
                System.out.println("relation attribute not
speficied");
                System.exit(-1);
            }
            double relation = drel.doubleValue();
            System.out.println("relation = " + relation);
            sum += relation * balance;
        }
    }
    return sum;
}
```

Appendix 6 gives the output of the execution of the validation performed in the FinancialStatement class for the invalid instance document txn.3.xml in Appendix 5. The balance sheet instance in txn3.xml is invalid (with invalid internal structure as discussed under the fifth component of validation) because the balances (200 and 80) of the netPropertyPlantEquipment and grossPropertyPlantEquipment elements are inconsistent with the sums (180 and 100) of the balances of its child elements. The output includes the listing of the DOMtree as well as the traversal of that tree for validation.

5 CONCLUDING OBSERVATIONS

Development of a language for business reporting is a long and arduous task. The XBRL specifications and the associated taxonomies have provided an initial statement of the requirements as well as the necessary infrastructure. In this paper we have utilized the XBRL C&I taxonomy to develop an application for one part of the reporting language: balance sheet. We have provided an elegant schema, shown it to be extensible, and provided a canonical structure for all financial

statements so that a standard algorithm can be developed for validation (primarily footing) of instance documents. We have discussed the algorithm as well as its implementation in java, and provided the test results.

Future research directions include the extension of the schema to all financial reports covered by C&I taxonomy including the notes to financial statements, the auditor's report, as well as meta-level information pertaining to financial statements. While our schema "looks" object-oriented in terms of the type hierarchies, the most important aspects of objects are absent (inheritance, information hiding/encapsulation). However, initial efforts are also under way to incorporate substantial object-oriented extensions.

6 REFERENCES

Berners-Lee, T; Hendler, J; Lassila, O.: *The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities.* http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html (visited on May 22, 2001)

DAML: *The DARPA Agent Markup Language Homepage (2000).* http://www.daml.org/

Kent, R.: *Conceptual Knowledge Markup Language: The Central Core.* http://sern.ucalgary.ca/ksi/kaw/kaw99/papers/Kent1/CKML.pdf, , (visited on May 22, 2001)

Sigel, A.: *Towards knowledge organization with Topic Maps.* http://www.infoloom.com/gcaconfs/WEB/paris2000/S22-02.HTM#s22-02shoeref1 (visited on May 22, 2001)

Vun Kannon, D.; Wang, Y.: *Design of the XBRL specification.* http://www.infoloom.com/gcaconfs/WEB/paris2000/S26-01.HTM (visited on June 6, 2001)

W3C (2000): *Extensible Markup Language (XML) 1.0 (Second Edition).* http://www.w3.org/TR/2000/REC-xml-20001006

W3C (2001a): *XML Schema Part 0: Prime.* http://www.w3.org/TR/xmlschema-0/ (visited on May 15, 2002)

W3C (2001b): *XML Schema Part 1: Structures.* http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/ (visited on May 15, 2002)

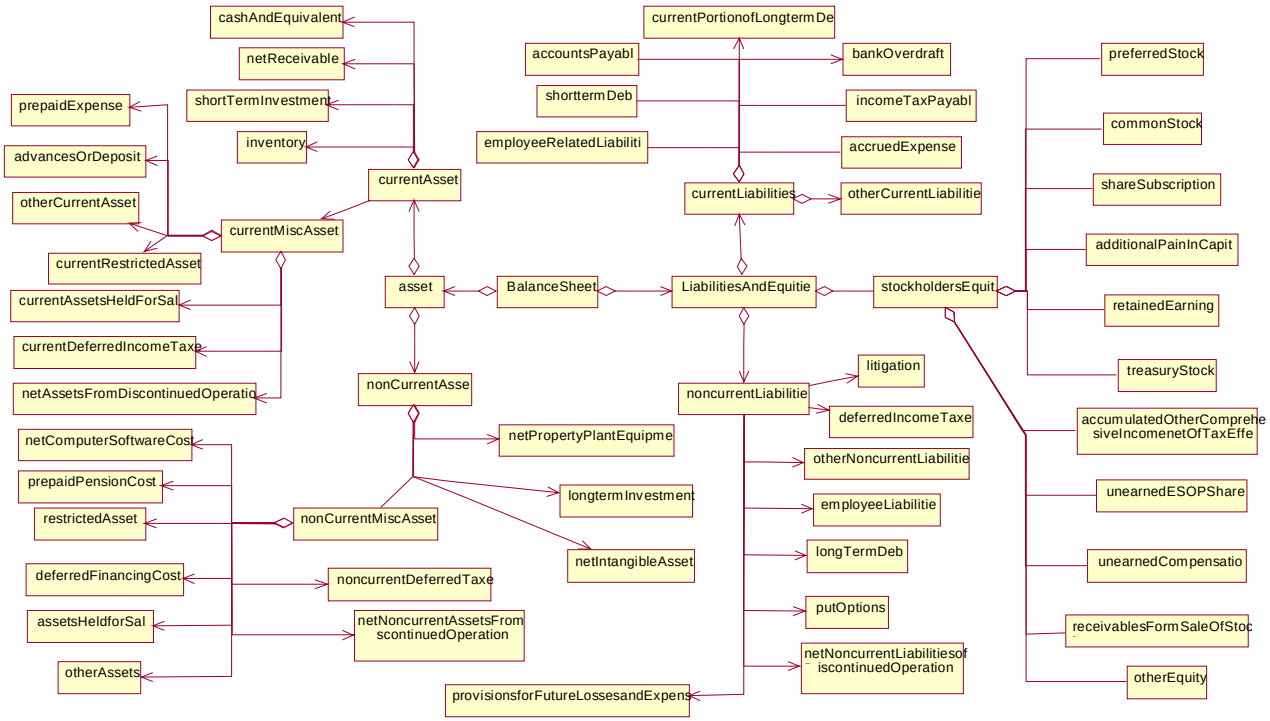
W3C (2001c): *XML Schema Part 2: Datatypes.* http://www.w3.org/TR/xmlschema-2/ (visited on May 15, 2002)

W3C (2001d): *XML Fragment Interchange.* http://www.w3.org/TR/xml-fragment (visited on May 15, 2002)

W3C (2002): *Resource Description Framework (RDF).* http://www.w3.org/RDF/ (visited on May 15, 2002)

W3C (2002):: Schema for Object-Oriented XML 2.0. http://www.w3.org/TR/NOTE-SOX (visited on May 15, 2002)

Appendix 1. The UML Class Diagram for Balance Sheet



Appendix 2. An XML-Schema for Balance Sheet (bsj6.xsd)

Note: *Ellipses -- sequence of dots (. . . . .) is used to improve readability*

```
<?xml version="1.0" encoding="UTF-8"?>
<!--The schema for Balance Sheet
Authors: Kinsun Tam, Sanjay Goel, and Jagdish S. Gangolly-->
<?xml-stylesheet type="text/xsl" href="bsj3.xsl"?>
<schema targetNamespace="http://www.albany.edu/acc/xfs"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xfs="http://www.albany.edu/acc/xfs"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

<!--
Definition of global attributes used in the schema.  The
attributes balance and relation are defined here.  The
attribute parent is defined locally and the value of that
attribute is bound to the name of the parent element

Balance is the amount as would appear on the Balance Sheet.
Relation is the nature of the relationship with the parent
element.
relation =  0 : the balance equals the balance of the parent
element
relation = +1 : the contribution of the element balance to
the balance of the parent element is positive
relation = -1 : the contribution of the element balance to
the balance of the parent element is negative
-->

   <!-- single out this netPropertyPlantEquipmentType
        for subsequent redefinition in imb4.xsd-->
   <complexType name="netPropertyPlantEquipmentType">
     <attribute name="relation" type="integer" fixed="+1"/>
     <attribute name="balance" type="decimal"
use="required"/>
     <attribute name="parent" type="NMTOKEN"
fixed="nonCurrentAssets"/>
   </complexType>
   <!-- Definitions for the types used in the hierarchy
        in the Balance Sheet.-->
   <!-- Definition of the currentAssetsType-->
   <complexType name="currentAssetsType">
     <all>
       <element name="cashAndEquivalents" minOccurs="0">
```

```
          <complexType>
            <attribute name="relation" type="integer"
fixed="+1"/>
            <attribute name="balance" type="decimal"
use="required"/>
            <attribute name="parent" type="NMTOKEN"
                 fixed="currentAssets"/>
            <!-- The parent attribute is defined locally.
                 currentAssets element is the parent of
                 cashAndEquivalents element-->
          </complexType>
        </element>
        <element name="netReceivables" minOccurs="0">
          <complexType>
            <attribute name="relation" type="integer"
fixed="+1"/>
            <attribute name="balance" type="decimal"
use="required"/>
            <attribute name="parent" type="NMTOKEN"
                 fixed="currentAssets"/>
          </complexType>
        </element>
        .....
     </all>
     <attribute name="relation" type="integer" fixed="+1"/>
     <attribute name="balance" type="decimal"
use="required"/>
     <attribute name="parent" type="NMTOKEN"
fixed="totalAssets"/>
   </complexType>
   <!-- Definition of the nonCurrentAssetsType-->
   <complexType name="nonCurrentAssetsType">
     <all>
       <element name="netPropertyPlantEquipment"
            type="xfs:netPropertyPlantEquipmentType"
minOccurs="0"/>
       <element name="longTermInvestments" minOccurs="0">
          <complexType>
            <attribute name="relation" type="integer"
fixed="+1"/>
            <attribute name="balance" type="decimal"
use="required"/>
            <attribute name="parent" type="NMTOKEN"
                 fixed="nonCurrentAssets"/>
          </complexType>
       </element>
       .....
```

```
     </all>
     <attribute name="relation" type="integer" fixed="+1"/>
     <attribute name="balance" type="decimal"
use="required"/>
     <attribute name="parent" type="NMTOKEN"
fixed="totalAssets"/>
   </complexType>
   <!-- Definition of the currentLiabilitiesType-->
   <complexType name="currentLiabilitiesType">
     <all>
       <element name="accountsPayable" minOccurs="0">
         <complexType>
           <attribute name="relation" type="integer"
fixed="+1"/>
           <attribute name="balance" type="decimal"
use="required"/>
           <attribute name="parent" type="NMTOKEN"
                fixed="currentLiabilities"/>
         </complexType>
       </element>
       <element name="shortTermDebt" minOccurs="0">
         <complexType>
           <attribute name="relation" type="integer"
fixed="+1"/>
           <attribute name="balance" type="decimal"
use="required"/>
           <attribute name="parent" type="NMTOKEN"
                fixed="currentLiabilities"/>
         </complexType>
       </element>
       .....
     </all>
     <attribute name="relation" type="integer" fixed="+1"/>
     <attribute name="balance" type="decimal"
use="required"/>
     <attribute name="parent" type="NMTOKEN"
          fixed="liabilitiesAndEquities"/>
   </complexType>
   <!-- Definition of the nonCurrentLiabilitiesType-->
   <complexType name="nonCurrentLiabilitiesType">
     <all>
       <element name="longTermDebt" minOccurs="0">
         <complexType>
           <attribute name="relation" type="integer"
fixed="+1"/>
           <attribute name="balance" type="decimal"
use="required"/>
```

```
            <attribute name="parent" type="NMTOKEN"
                 fixed="nonCurrentLiabilities"/>
          </complexType>
        </element>
        <element name="deferredIncomeTaxes" minOccurs="0">
          <complexType>
            <attribute name="relation" type="integer"
fixed="+1"/>
            <attribute name="balance" type="decimal"
use="required"/>
            <attribute name="parent" type="NMTOKEN"
                 fixed="nonCurrentLiabilities"/>
          </complexType>
        </element>
        .....
     </all>
     <attribute name="relation" type="integer" fixed="+1"/>
     <attribute name="balance" type="decimal"
use="required"/>
     <attribute name="parent" type="NMTOKEN"
          fixed="liabilitiesAndEquities"/>
  </complexType>
  <!-- Definition of the stockholdersEquityType-->
  <complexType name="stockholdersEquityType">
     <all>
        <element name="preferredStock" minOccurs="0">
          <complexType>
            <attribute name="relation" type="integer"
fixed="+1"/>
            <attribute name="balance" type="decimal"
use="required"/>
            <attribute name="parent" type="NMTOKEN"
                 fixed="stockholdersEquity"/>
          </complexType>
        </element>
        <element name="commonStock" minOccurs="0">
          <complexType>
            <attribute name="relation" type="integer"
fixed="+1"/>
            <attribute name="balance" type="decimal"
use="required"/>
            <attribute name="parent" type="NMTOKEN"
                 fixed="stockholdersEquity"/>
          </complexType>
        </element>
        .....
     </all>
```

```
     <attribute name="relation" type="integer" fixed="+1"/>
     <attribute name="balance" type="decimal"
use="required"/>
     <attribute name="parent" type="NMTOKEN"
          fixed="liabilitiesAndEquities"/>
   </complexType>
   <!-- Definition of the liabilitiesAndEquitiesType-->
   <complexType name="liabilitiesAndEquitiesType">
     <all>
       <element name="currentLiabilities"
            type="xfs:currentLiabilitiesType"
minOccurs="0"/>
       <element name="nonCurrentLiabilities"
            type="xfs:nonCurrentLiabilitiesType"
minOccurs="0"/>
       <element name="stockholdersEquity"
            type="xfs:stockholdersEquityType"
minOccurs="0"/>
     </all>
     <attribute name="relation" type="integer" fixed="0"/>
     <attribute name="balance" type="decimal"
use="required"/>
     <attribute name="parent" type="NMTOKEN"
fixed="balanceSheet"/>
   </complexType>
   <!-- Definition of the totalAssetsType-->
   <complexType name="totalAssetsType">
     <all>
       <element name="currentAssets"
type="xfs:currentAssetsType"
            minOccurs="0"/>
       <element name="nonCurrentAssets"
type="xfs:nonCurrentAssetsType"
            minOccurs="0"/>
     </all>
     <attribute name="relation" type="integer" fixed="0"/>
     <attribute name="balance" type="decimal"
use="required"/>
     <attribute name="parent" type="NMTOKEN"
fixed="balanceSheet"/>
   </complexType>
   <!-- Definition of the balanceSheetType-->
   <complexType name="balanceSheetType">
     <all>
       <element name="totalAssets"
type="xfs:totalAssetsType"
            minOccurs="0"/>
```

```
        <element name="liabilitiesAndEquities"
             type="xfs:liabilitiesAndEquitiesType"
minOccurs="0"/>
     </all>
     <attribute name="relation" type="integer" fixed="0"/>
     <attribute name="balance" type="decimal"
use="required"/>
     <attribute name="parent" type="NMTOKEN"
fixed="statement"/>
   </complexType>
   <!-- Definition of the root element balanceSheet-->
   <element name="balanceSheet" type="xfs:balanceSheetType"/>
</schema>
```

Appendix 3. A Schema Extension Example for Balance Sheet (txn2.xsd)

Note: *Ellipses -- sequence of dots (. . . . .) is used to improve readability*

```
<?xml version="1.0" encoding="UTF-8"?>
<!--The schema for Balance Sheet extension
Authors: Kinsun Tam, Sanjay Goel, and Jagdish S. Gangolly-->
<?xml-stylesheet type="text/xsl" href="bsj3.xsl"?>
<schema targetNamespace="http://www.albany.edu/acc/xfs"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xfs="http://www.albany.edu/acc/xfs"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

   <redefine schemaLocation="bsj6.xsd">
      <complexType name="netPropertyPlantEquipmentType">
         <complexContent>
            <extension
base="xfs:netPropertyPlantEquipmentType">
               <all>
                  <!-- Definition of the
grossPropertyPlantEquipment-->
                  <element name="grossPropertyPlantEquipment"
                       minOccurs="0">
                     <complexType>
                        <all>
                           <element name="land" minOccurs="0">
                              <complexType>
                                 <attribute name="relation"
type="integer"
                                      fixed="+1"/>
                                 <attribute name="balance"
type="decimal"
                                      use="required"/>
                                 <attribute name="parent"
type="NMTOKEN"

   fixed="grossPropertyPlantEquipment"/>
                              </complexType>
                           </element>
                           <element name="machineryAndEquipment"
                                minOccurs="0">
                              <complexType>
                                 <attribute name="relation"
type="integer"
                                      fixed="+1"/>
```

```
                              <attribute name="balance"
type="decimal"
                                   use="required"/>
                              <attribute name="parent"
type="NMTOKEN"

   fixed="grossPropertyPlantEquipment"/>
                              </complexType>
                           </element>
                           .....
                        </all>
                        <attribute name="relation" type="integer"
                             fixed="+1"/>
                        <attribute name="balance" type="decimal"
                             use="required"/>
                        <attribute name="parent" type="NMTOKEN"
                             fixed="netPropertyPlantEquipment"/>
                     </complexType>
                  </element>
                  <!-- Definition of the
                       accumulatedDepreciationAndAmortization-->
                  <element
name="accumulatedDepreciationAndAmortization"
                       minOccurs="0">
                     <complexType>
                        <attribute name="relation" type="integer"
                             fixed="-1"/>
                        <attribute name="balance" type="decimal"
                             use="required"/>
                        <attribute name="parent" type="NMTOKEN"
                             fixed="netPropertyPlantEquipment"/>
                     </complexType>
                  </element>
                  .....
               </all>
            </extension>
         </complexContent>
      </complexType>
   </redefine>
</schema>
```

Appendix 4. An XSL Stylesheet for Balance Sheet (bsj3.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
xmlns:xfs="http://www.albany.edu/acc/xfs" version="1.0">
   <xsl:template match="/">
      <html>
         <head>
            <title>   Displaying XML Balance Sheet with XSL
   </title>
         </head>
         <body bgcolor="#666666" text="#FFFFFF"
display="block">
            <font size="+2"/>
               Balance Sheet
               <hr/>
            <xsl:apply-templates/>
         </body> </html>
   </xsl:template>
   <xsl:template match="xfs:balanceSheet">
      <xsl:for-each select="*">
         <xsl:node-name/>
         <font color="FFCC33">
            <xsl:value-of select="@xfs:balance"/>
         </font> <br/>
         <xsl:for-each select="*">
            <ul>
            <xsl:node-name/>
            <font color="FFFFCC">
               <xsl:value-of select="@xfs:balance"/>
            </font>
            <br/>
            <xsl:for-each select="*">
               <ul>
               <xsl:node-name/>
               <font color="FFFFCC">
                  <xsl:value-of select="@xfs:balance"/>
               </font> <br/>
               <xsl:for-each select="*">
                  <ul>
                  <xsl:node-name/>
                  <font color="FFFFCC">
                     <xsl:value-of select="@xfs:balance"/>
                  </font>
                  <br/>
```

```
                    <xsl:for-each select="*">
                    <ul>
                    <xsl:node-name/>
                    <font color="FFFFCC">
                      <xsl:value-of select="@xfs:balance"/>
                    </font>
                    <br/>
                    <xsl:for-each select="*">
                      <ul>
                      <xsl:node-name/>
                      <font color="FFFFCC">
                        <xsl:value-of select="@xfs:balance"/>
                      </font>
                      <br/>
                      </ul>
                    </xsl:for-each>
                    </ul>
                  </xsl:for-each>
                  </ul>
                </xsl:for-each>
                </ul>
              </xsl:for-each>
              </ul>
            </xsl:for-each>
          </xsl:for-each>
        </xsl:for-each>
      </xsl:template>
    </xsl:stylesheet>
```

Appendix 5. An Instance of a Balance Sheet with a Schema Extension (txn3.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Balance Sheet Instance
Authors: Kinsun Tam, Sanjay Goel, and Jagdish S. Gangolly-->
<?xml-stylesheet type="text/xsl" href="bsj3.xsl"?>

<balanceSheet balance="600"
xmlns="http://www.albany.edu/acc/xfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.albany.edu/acc/xfs txn2.xsd">
   <totalAssets balance="600.00">
      <currentAssets balance="100.00">
         <cashAndEquivalents balance="10.00"/>
         <netReceivables balance="20.00"/>
         <shortTermInvestments balance="0.00"/>
         <inventory balance="40.00"/>
         <currentMiscellaneousAssets balance="30.00"/>
      </currentAssets>
      <nonCurrentAssets balance="500.00">
         <netPropertyPlantEquipment balance="200.00">
            <grossPropertyPlantEquipment balance="80.00">
               <land balance="40.00"/>
               <machineryAndEquipment balance="30.00"/>
               <furnitureAndFixtures balance="20.00"/>
               <computerEquipment balance="10.00"/>
            </grossPropertyPlantEquipment>
            <accumulatedDepreciationAndAmortization
balance="150.00"/>
            <netCapitalLeaseAssets balance="250.00"/>
         </netPropertyPlantEquipment>
         <longTermInvestments balance="100.00"/>
         <netIntangibleAssets balance="100.00"/>
         <nonCurrentMiscellaneousAssets balance="100.00"/>
      </nonCurrentAssets>
   </totalAssets>
   <liabilitiesAndEquities balance="600.00">
      <currentLiabilities balance="200.00">
         <accountsPayable balance="10.00"/>
         <shortTermDebt balance="20.00"/>
         <currentPortionOfLongTermDebt balance="30.00"/>
         <bankOverdrafts balance="40.00"/>
         <incomeTaxPayable balance="50.00"/>
         <accruedExpenses balance="50.00"/>
```

```
        <employeeRelatedLiabilities balance="0.00"/>
      </currentLiabilities>
      <nonCurrentLiabilities balance="100.00">
        <longTermDebt balance="10.00"/>
        <deferredIncomeTaxes balance="10.00"/>
        <litigation balance="10.00"/>
        <employeeLiabilities balance="20.00"/>
        <putOptions balance="20.00"/>
        <netNonCurrentLiabilitiesOfDiscontinuedOperations
balance="0"/>
        <provisionForFutureLossesAndExpenses
balance="30.00"/>
      </nonCurrentLiabilities>
      <stockholdersEquity balance="300.00">
        <preferredStock balance="100.00"/>
        <commonStock balance="100"/>
        <shareSubscriptions balance="30"/>
        <additionalPaidInCapital balance="70"/>
        <retainedEarnings balance="100"/>
        <treasuryStock balance="100"/>
      </stockholdersEquity>
    </liabilitiesAndEquities>
</balanceSheet>
```

Appendix 6. Output of the Validation Program Applied on txn3.xml

Note: *Ellipsis -- sequence of dots (. . . . .) is used to improve readability. A bug in DOM is erased.*

```
Getting the root node
Testing the validity of the document
checkCascadingBalance LEVEL 1
Node Name balanceSheet
Number of Children = 2
Node balance parent = 600.0
Node balance child = 600.0
Node balance child = 600.0
Boolean from checkEqualityRelation = true


checkCascadingBalance LEVEL 2
Node Name totalAssets
Number of Children = 2
Parent Node Balance = 600.0
Printing child node balance
balance = 100.0
relation = 1.0
balance = 500.0
relation = 1.0
Children node balance = 600.0
 Difference = 0.0
.....
Name = currentAssets
Name = nonCurrentAssets


checkCascadingBalance LEVEL 3
Node Name currentAssets
Number of Children = 5
Parent Node Balance = 100.0
Printing child node balance
balance = 10.0
relation = 1.0
balance = 20.0
relation = 1.0
balance = 0.0
relation = 1.0
balance = 40.0
relation = 1.0
balance = 30.0
relation = 1.0
```

```
Children node balance = 100.0
 Difference = 0.0
.....



Name = cashAndEquivalents
Name = netReceivables
Name = shortTermInvestments
Name = inventory
Name = currentMiscellaneousAssets



checkCascadingBalance LEVEL 4
Node Name cashAndEquivalents
Reached Leaf Node
checkCascadingBalance LEVEL 4
Node Name netReceivables
Reached Leaf Node
checkCascadingBalance LEVEL 4
Node Name shortTermInvestments
Reached Leaf Node
checkCascadingBalance LEVEL 4
Node Name inventory
Reached Leaf Node
checkCascadingBalance LEVEL 4
Node Name currentMiscellaneousAssets
Reached Leaf Node


checkCascadingBalance LEVEL 3
Node Name nonCurrentAssets
Number of Children = 4
Parent Node Balance = 500.0
Printing child node balance
balance = 200.0
relation = 1.0
balance = 100.0
relation = 1.0
balance = 100.0
relation = 1.0
balance = 100.0
relation = 1.0
Children node balance = 500.0
 Difference = 0.0
```

```
.....
Name = netPropertyPlantEquipment
Name = longTermInvestments
Name = netIntangibleAssets
Name = nonCurrentMiscellaneousAssets




checkCascadingBalance LEVEL 4
Node Name netPropertyPlantEquipment
Number of Children = 3
Parent Node Balance = 200.0
Printing child node balance
balance = 80.0
relation = 1.0
balance = 150.0
relation = -1.0
balance = 250.0
relation = 1.0
Children node balance = 180.0
 Difference = 20.0
.....
Name = grossPropertyPlantEquipment
Name = accumulatedDepreciationAndAmortization
Name = netCapitalLeaseAssets
.....The document file:c:\xbrlproject\tam\tamsrc\txn3.xml is
not valid.
```