The International Journal of Digital Accounting Research
Vol. 3, No. 6, 2003, pp. 133-164
ISSN: 1577-8517

An XML Schema Design Framework to Simplify Financial Statement Validation[1]

Kinsun Tam. University at Albany, State University of New York. U.S.A.
tam@albany.edu

Abstract. Financial statements share a common directed tree model. This common structure simplifies validation of financial statement instance documents and has important implications for the design of business reporting markup languages. Exploiting this a common structure enables different financial statements to share the same validation routine, thereby realizing considerable savings in development and maintenance of validation applications. This paper derives a common schema for financial statements based on a standard directed tree model, and demonstrates its contributions towards standardizing and simplifying validation of financial statements. This efficient framework of taxonomy development can be applied to other domains because many non-financial business documents also have a hierarchical structure expressible as a directed tree.

Key words: Extensible Markup Language; Business Reporting Language; Validation of Financial Reports; Directed Tree Model.

1. INTRODUCTION

Validation of XML financial statement instances prior to their display in browsers is critical in providing assurance on reliability of their information structure and semantics. Such validation also minimizes costs of processing invalid data received electronically. This emphasis on validation has gained considerable notice in deliberations by standard setters in the design of taxonomies for XML

[1] The author gratefully acknowledges research support from the State University of New York at Albany Faculty Research Award Program and from the State University of New York at Albany School of Business.

derivatives. For example, proposals for validation specifications are now being solicited for Financial Product Markup Language (FpML). In addition, a major objective of Extensible Business Reporting Language (XBRL) Specification (Version 2.0) is to embrace XML schema-based validation (W3C, 2001a, 2001b, and 2001c).

The economic significance of this study can be appreciated from the large number of countries developing and implementing XML-based taxonomy for business reporting (Vasal and Srivastava, 2002). An XML schema design framework that optimizes performance, therefore, will promote efficiency on a global scale.

Prior research has studied implications of validation on the design of XML taxonomies for business reporting. Leuder (2000) and Tam *et al.* (2002) suggest that modeling financial statement items as hierarchical structures can simplify validation. Using the balance sheet as an example, both studies point to easier validation being achieved by exploiting XML's strength in representing hierarchical data structures.

This paper extends the above stream of research. Leveraging XML's strength in representing hierarchical data simplifies financial statement validation. This advantage applies to all financial statements (with the possible exception of footnotes) because they are all inherently hierarchical. This study demonstrates that a common XML schema design framework can be applied to all financial statements so that a single algorithm can validate all financial statement instances.

In particular, the income statement, the statement of cash flows, the statement of comprehensive income, and the statement of stockholders' equity all share a hierarchical structure similar to that of the balance sheet. XML schemas for these statements therefore can be developed based on the same design criteria as for the balance sheet. Tam *et al.* (2002) illustrate that since the balance sheet is expressible as a directed tree, a tree-traversal validation algorithm together with a generic XML parser can validate business rules peculiar to the balance sheet[2]. This study generalizes their result to show that all financial statements can be expressed as a directed tree, and therefore can be validated by the same tree-traversal validation algorithm against respective financial statement rules.

---

[2] A directed tree is a set of nodes connected with directed edges without forming circles.

Since users rely on financial statements delivered through the Internet, validation of financial statements (at the application level) against financial statement rules is imperative. Examples of such rules include «Total cash flows = operating cash flows + financing cash flows + investment cash flows» and «Gross margin = sales - CGS». Since this study exploits the common directed tree model for financial statements, a common validation algorithm can validate all these statements against their respective rules. This is a significant advantage because validation routines need not be separately developed and maintained. Systems at the receiving end of a particular financial statement need not search for a matching routine.

This research is motivated by deficiency of existing taxonomies and the possibility of building object models to guide taxonomy development. A primary objective of this research is to demonstrate how a common XML schema design framework that optimizes performance can guide taxonomy-building decisions for various financial statements. Without a sound theory to guide design decisions, expedience rather than efficiency may dominate the design process, resulting in a costly end product.

By suggesting a common taxonomy design for all financial statements, this research holds potential of streamlining financial statement validation and hence financial reporting through the Internet. In addition, because many other (non financial) business documents are essentially hierarchical in structure, this research may provide a more efficient framework of XML taxonomy development for all other industries (W3C, 2000). Efficient taxonomies can expedite B2B communications and e-commerce, and may reduce the likelihood of errors in financial statement instances.

Research on taxonomy design provides insights for XBRL development. XBRL Specification Version 2 is making improvements in representing the hierarchical structure of financial statements. Version 2, like its predecessor, disallows nesting of XBRL item elements, but permits structural relationships to be captured in tuples and in link structures (XBRL, 2000, 2001). These changes enable structural information necessary for validation to be documented more accurately. However, XBRL Version 2 still does not directly express aggregation relationships as nesting hierarchies. XBRL Version 2 places «assets» and «current assets» at the same hierarchical level in an instance document, obscuring the fact that «current assets» is actually an account within «assets». This makes XBRL instance documents less readable, and their validation more difficult because they provide no information about hierarchical structure to the validating software.

Remaining sections are organized as follows. Section 2 reviews relevant prior literature and motivates the current study. Section 3 discusses implications of schema design on validation efficiency. In Section 4, the directed tree model of financial statements is described. Section 5 provides examples in XML of the common directed tree structure, and demonstrates validation of various kinds of financial statement by one generic program. Section 6 provides concluding observations.

2. PRIOR LITERATURE AND MOTIVATION

According to Sun Microsystems (1994), a major contributor to early stage XML development, XML promises to strengthen data portability across different systems and lower the cost of data interchange. Not surprisingly, commercial implementations of XML are primarily for business data exchange and enterprise application integration purposes (Giga Information Group, 2001; GAO, 2002). For instance, Bonson (2001) highlights the role of XBRL (an XML derivative) in enabling machines to understand the accounting language to facilitate automatic exchange of information between softbots and/or software applications.

To assure accuracy, accounting information exchanged between machines needs to be monitored by validation software. Data accuracy and validation are major considerations in applying XML towards business reporting. Hoffman *et al*. (1999) explain XML's advantages in facilitating error-free conveyance of complex business information, and enabling validation software to reject nonconforming data. Looking ahead beyond validation, Bovee *et al.* (2001) and Vasal and Srivastava (2002) foresee future roles of XML and XBRL in continuous monitoring and auditing of business information.

Lowering the cost of business data interchange requires efficient validation routines. Leuder (2000) and Tam *et al.* (2002) argue for exploiting the hierarchical structure of financial statements to simplify the validation process. Leuder (2000) has discussed limitations in XBRL Version 1's design. In particular, XBRL Version 1 does not exploit the strength of XML in representing structural data, does not

---

[3] XBRL Version 1 defines only the group and item element tags.

directly express aggregation relationships as nesting hierarchies, and does not define domain-specific XML elements to represent financial statement items[3]. These deviations from the common approach of XML extension render XBRL incompatible with generic XML validating parsers, and make it necessary to separately specify hierarchy nesting. Despite changes in XBRL Version 2.0 to embrace XML schema-based validation, Version 2 still does not directly express aggregation relationships as nesting hierarchies[4].

Tam *et al.* (2002) discusses desirable design criteria for a business reporting markup language. These criteria include use of standard modeling methods, exploitation of common schema structure for all financial statements, use of meaningful domain-specific tags, enhancement of XML code readability, enabling extensibility of schema, promotion of modularity, and payload overhead minimization. As a proof-of-concept, they provided the design of an XML schema for the balance sheet, which illustrates these criteria.

Advantages of all abovementioned design criteria, with one exception, are evident from Tam *et al.*'s (2002) balance sheet example. The second criterion, calling for exploiting the common structure underlying all financial statements, is an important advantage of their proposed design. This criterion is proposed and argued for, but its feasibility and advantages are not demonstrated in their proof-of-concept example that focuses only on the balance sheet.

The key objective of this paper is to provide evidence to support the second criterion, and highlight its importance in the design of markup languages. Financial statements share a common directed tree structure of hierarchies, based on which a single generic tree-traversal algorithm for validation can be programmed.

While hierarchy-based modeling takes advantage of generic XML validating parsers, exploiting the common directed tree model of financial statements enables a common algorithm to be applied to all financial statement instance documents, thus further simplifying validation.

---

[4] In XBRL Version 2, structural and hierarchy nesting information needs to be separately stored in tuples and linkbases.

3. IMPLICATION OF SCHEMA DESIGN ON VALIDATION EFFICIENCY

Validation of XML instance documents can mean many things. Validating schemas or schema extensions (or DTD and DTD extensions) against XML schema specification (or DTD specification), validating of stylesheets against XSL or CSS specifications, and validating financial statement instances against schema or DTD are generic to all XML documents and can be performed at the browser level.

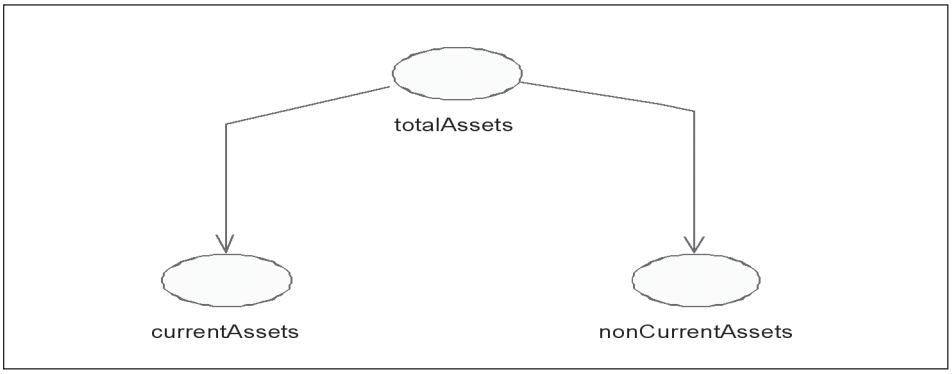
In contrast, validation with respect to semantic relationships driven by rules from the accounting domain is unique to XML financial statement instances. Because data in financial statement instances are used for investment decisions, this validation is required to provide assurance to financial statement readers. While it is appropriate to speak of validation as an application level concern, browser applications (Internet Explorer, Netscape, etc.) are unlikely to incorporate necessary knowledge for such domain-specific validation. Accordingly, semantics of accounting rules must be separately programmed.

From an efficiency standpoint, it is important to consider implications of schema design on the programming of this domain-specific validation routine. By exploiting the common schema structure across all financial statements, it is possible to standardize validation routines for all financial statements. Validation routine for each kind of financial statement need not be separately developed and maintained. The system at the receiving end of financial statements need not search for a particular matching validation program. Tam *et al.* (2002) summarize this argument as follows:

> «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.»

4. A COMMON DIRECTED TREE MODEL FOR FINANCIAL STATEMENTS

Modeling financial statements as a directed tree enables financial statement rules and hierarchical relationships to be specified in the tree structure. Because both the directed tree model and the XML schema are hierarchical, translation from a directed tree model to an XML schema is intuitive. For instance, a directed tree depicting aggregation relationship between Total Assets and its constituents (Current Assets and Noncurrent Assets) is specified as follows:



This aggregation relationship can be translated into XML as a hierarchy of XML elements. The XML hierarchy below identifies «`currentAssets`» and «`nonCurrentAssets`» as child elements of «`totalAssets`»:

```
<totalAssets>
       <currentAssets> ... </currentAssets>
       <nonCurrentAssets> ... </nonCurrentAssets>
</totalAssets>
```

Alternatively, if the above aggregation relationship is not expressed as a hierarchy of XML elements, elements will appear as unrelated (see below). In this case, relationships information between seemingly unrelated elements, if not separately captured elsewhere, will be lost.

```
<totalAssets> ... </totalAssets>
<currentAssets> ... </currentAssets>
<nonCurrentAssets> ... </nonCurrentAssets>
```

The balance sheet can be represented as a directed tree with multiple hierarchical layers. The basic building block of the directed tree model is described in Figure 1. The design is primitive and parsimonious because it is intended for concept-illustration rather than production purpose.

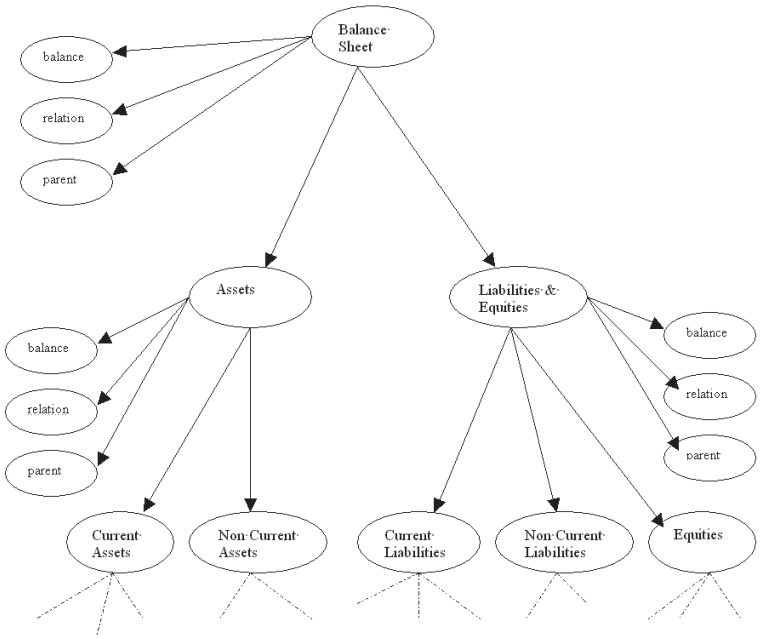
It consists of an account item element (such as Assets or Liabilities and Equities) and attributes (balance, parent, and relation). The `balance` attribute stores the monetary value of the element. The `parent` attribute identifies the parent node of the current element. The `relation` attribute is used to specify semantic relationships with respect to rules from accounting domain.

Depending on relationship with the «parent» element, the `relation` attribute may assume the value of +1, 0, or -1. When an element's `balance` contributes positively to its parent's `balance` (e.g. an increase in cash causes a corresponding increase in current assets), the `relation` attribute takes the value of +1. When the `relation` value is 0, siblings are not supposed to be aggregated. For instance, two child elements of «`commonStock`», namely «`commonStockDescription`» and «`shares-Authorized`», are not to be aggregated because of different units of measure[5]. Finally, a -1 value of the `relation` attribute indicates negative contribution to the parent's balance (as in the case of `treasuryStocks'` contribution to `stockholdersEquity`).

By allowing an element (such as `assets`) to have child elements (such as `currentAssets`), a directed tree consisting of a hierarchy of accounts can be built to represent any financial statement. Translation from the directed tree model to XML schema, owing to their common hierarchical structure, is intuitive. Figure 2 presents the directed tree model of the balance sheet in UML.

---

[5] This example is taken from US GAAP Commercial and Industrial (Presentation Report) (XBRL, 2002).



Note: The `balance` attribute stores the value of the element. The `parent` attribute identifies the parent of the current element. The `relation` attribute keeps track of the relationship with the «parent» element as follows:

`relation` = 0 : the «`balance`» does not contribute to the parent element This occurs when the siblings are not supposed to be aggregated. In `incomeStatement`, for instance, `netIncomeAvailableTo-Common` and `earningsPerShareInformation` have different units of measure, and are not to be aggregated[6]

`relation` = +1 : the contribution of «`balance`» to the balance of the parent element is positive

`relation` = -1 : the contribution of «`balance`» to the balance of the parent element is negative

Figure 1. The Basic Building Block of the Directed Tree Model of Financial Statements

[6] An element such as earningsPerShareInformation probably should not have relation and balance attributes. As earningsPerShareInformation is defined as an element of incomeStatement under the XBRL US GAAP C&I Taxonomy, it is implemented just like any other child element of incomeStatement for the sake of consistency in programming. An alternative implementation without the relation and balance attributes has been attempted and found to be feasible.

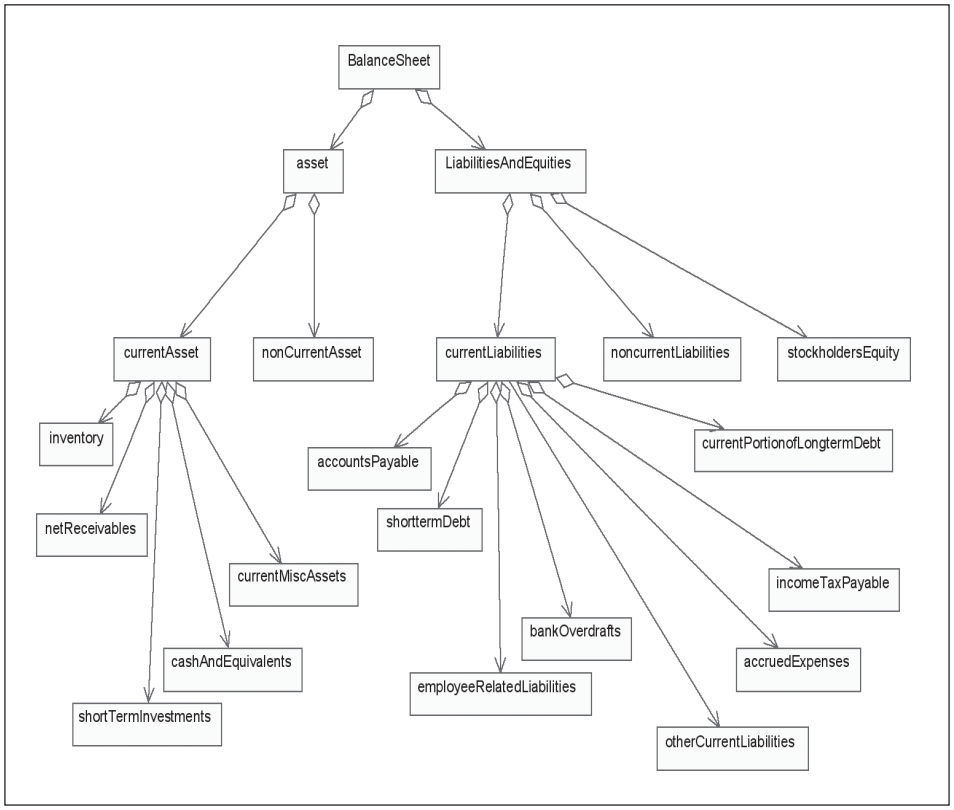


Figure 2. UML Class Diagram for Balance Sheet (partial). Arrows represent aggregation.

5. VALIDATING FINANCIAL STATEMENTS SHARING A COMMON MODEL

5.1. A common directed tree model in UML for all financial statements

In fact, all common financial statements have a hierarchical structure that can be expressed in a directed tree model. Figure 3 depicts directed tree models of the income statement, the cash flows statement, the statement of comprehensive income, and the statement of stockholders' equity in UML class diagram. There are noticeable individual features in these figures. For instance, the income statement is the least bushy, and the statement of stockholders' equity is the shortest (with the smallest number of levels). Despite all these individual characteristics, however, each statement exhibits a directed tree structure.

The income statement can be expressed as a directed tree (Figure 3, Panel A) with «`incomeStatement`» as the root element, and «`netIncome-AvailableToCommon`» and «`earningsPerShareInformation`» as child elements. Other components of the income statement, such as «`netIncome`» and «`perferredDividends`» extend this directed tree with additional hierarchies. Likewise, the cash flows statement (Figure 3, Panel B), the statement of comprehensive income (Figure 3, Panel C), and the statement of stockholders' equity (Figure 3, Panel D) are expressible as directed trees with root elements («`cashFlowStatement`», «`statementOfComprehensiveIncome`», and «`statementOfStockholdersEquity`») and lower level hierarchies.

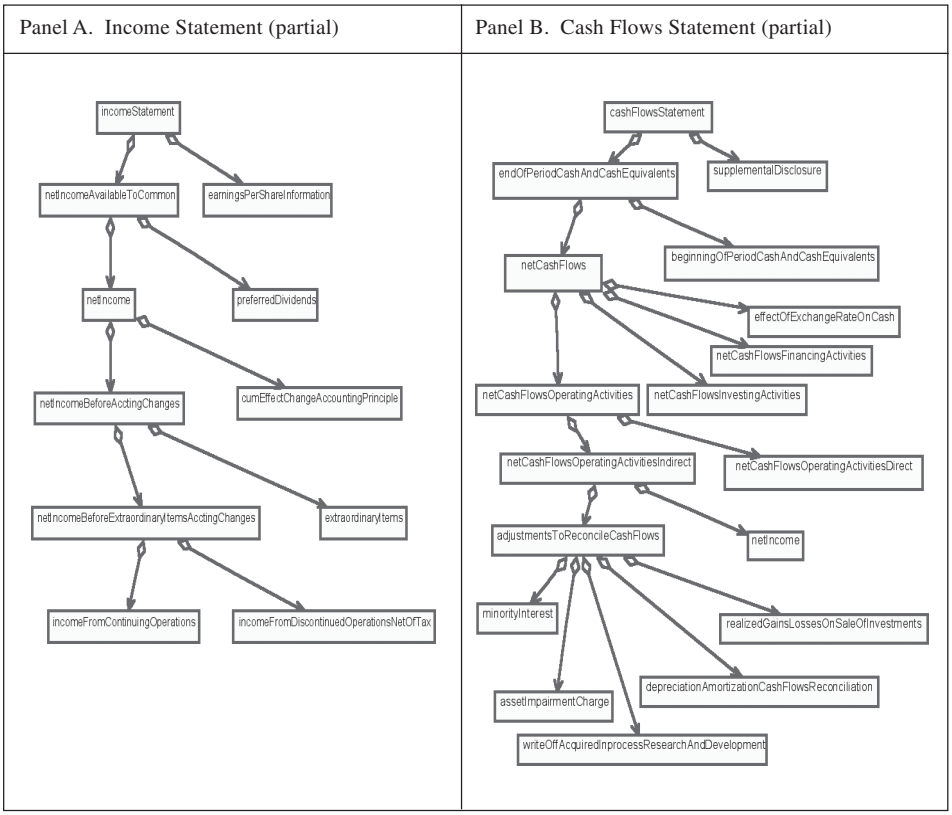


Figure 3. UML Class Diagram for Income Statement (partial), Cash Flows Statement (partial), Statement of Comprehensive Income, and Statement of Stockholders' Equity (partial). Arrows represent aggregation.

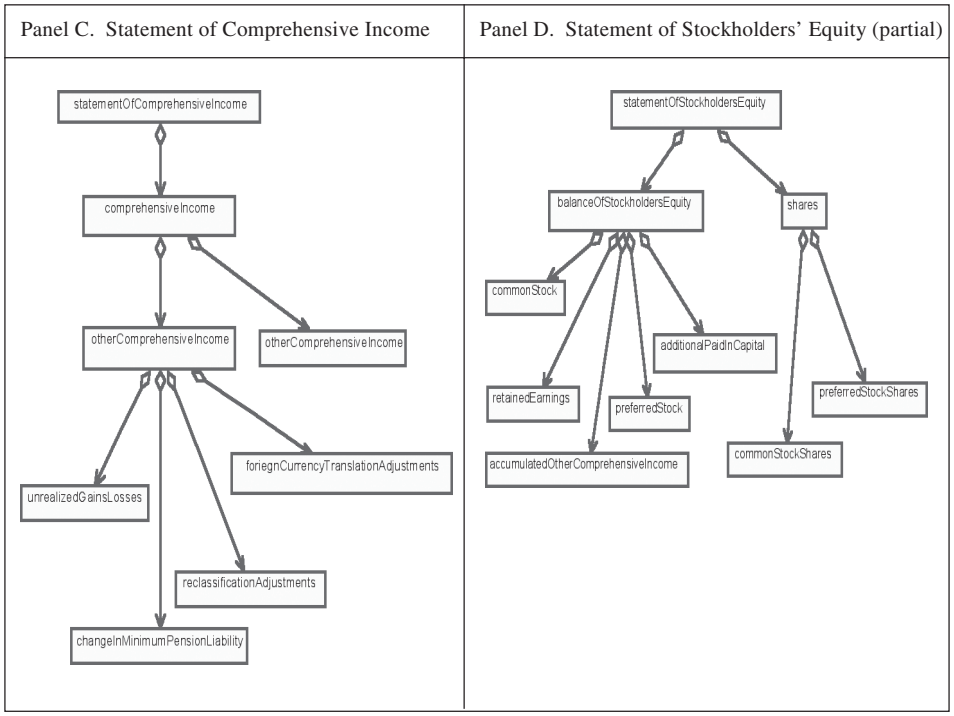


Figure 3 (Continued). UML Class Diagram for Income Statement (partial), Cash Flows Statement (partial), Statement of Comprehensive Income, and Statement of Stockholders' Equity (partial). Arrows represent aggregation.

This common directed tree structure, if appropriately taken into account in the design of markup languages, can simplify domain-specific validation. In particular, only one generic algorithm to traverse a tree and to validate relationships between nodes is needed. Once developed, this generic routine can then be used to validate any financial statement instance document with a directed tree structure.

5.2. A common algorithm for validating all financial statements

The following subsections provide examples of XML schema and instance documents for various financial statements, and demonstrate how a single generic program is sufficient to validate all financial statement instance documents against their respective schemas. As a proof-of-concept, a Java-based validation program for this purpose has been developed. UML class diagrams in Figure 4 illustrate components of this program.

The Default Package (Figure 4 Panel A) contains the five core Java classes (i.e. FinancialStatement, Util, Parse, DOMErrorHandler, and Validate) of the validation program. This package depends on the java.io, java.util, and java.lang packages provided by Sun Microsystems' Java 2 Platform and other XML parsing and transformation packages provided by the Apache Software Foundation's XML Project[7]. Dependency in UML diagrams is represented by a dashed line pointing to the package being depended upon[8].

Execution of the validation program starts with the «`main()`» method of the FinancialStatement class, which invokes methods of Parse, Util, and Validate classes. Methods of the Parse class (Figure 4 Panel C) help create a document tree out of the financial statement instance document to be validated. The «`getRootNode()`» method of the FinancialStatement class gets to the root of this document tree. The Validate class (Figure 4 Panel D) contains methods to access the `balance` and `relation` attributes of a node, and to recursively check every node on the document tree against relevant financial statement rules. The DOMErrorHandler class (Figure 4 Panel E) and the Util class (Figure 4 Panel F) provide definitions and methods to support parsing.

---

[7] Packages provided by the Apache Software Foundation includes javax.xml.parsers, javax.xml.transform, javax.xml.transform.stream, javax.xml.transformation.dom, org.w3c.dom, org.xml.sax.helpers, and org.xml.sax. Sun's Java 2 Platform (Standard Edition, v 1.3.1), on which the validation program of this study is compiled, does not include these packages. However, Sun has incorporated the Apache Software Foundation's packages into Java 2 Platform v 1.4.0.

This program uses Apache Software Foundation's Xerces 1.4 Parser. The Apache Software Foundation's Xerces2 Java Parser 2.0.2 has been available since June 2002. Xerces2 is meant to support eXtensible Markup Language (XML) 1.0 Second Edition Recommendation and related API's for XML processing. However, at the time of writing this paper, Xerces2 is still not steady. Therefore, this study reverts to using Xerces Java Parser 1.4.0.

The use of Xerces Java Parser 1.4.0 causes accommodations in the design of XML schema for financial statements. Xerces 1.4.0 does not implement infoset contribution correctly. Only attributes with unqualified (unprefixed) names are added as infoset contribution. Accordingly, this study has to use simple types for relation and balance. This causes relation and balance to become non-root level attributes, for which xfs prefix is not needed in instance document. Therefore, the xmlns:xsf definition is removed from xml instance document. Finally, xfs prefix is removed from element names in instance document. Nevertheless, these accommodations, while making the schema design less elegant, does not affect the central argument of this paper that, when the markup language design takes into account the common directed tree structure of financial statements, a single validation program suffices to perform validation of all financial statement instance documents.

[8] In UML, dependency refers to a relationship where the semantic characteristics of one entity rely upon the semantic characteristics of another entity.

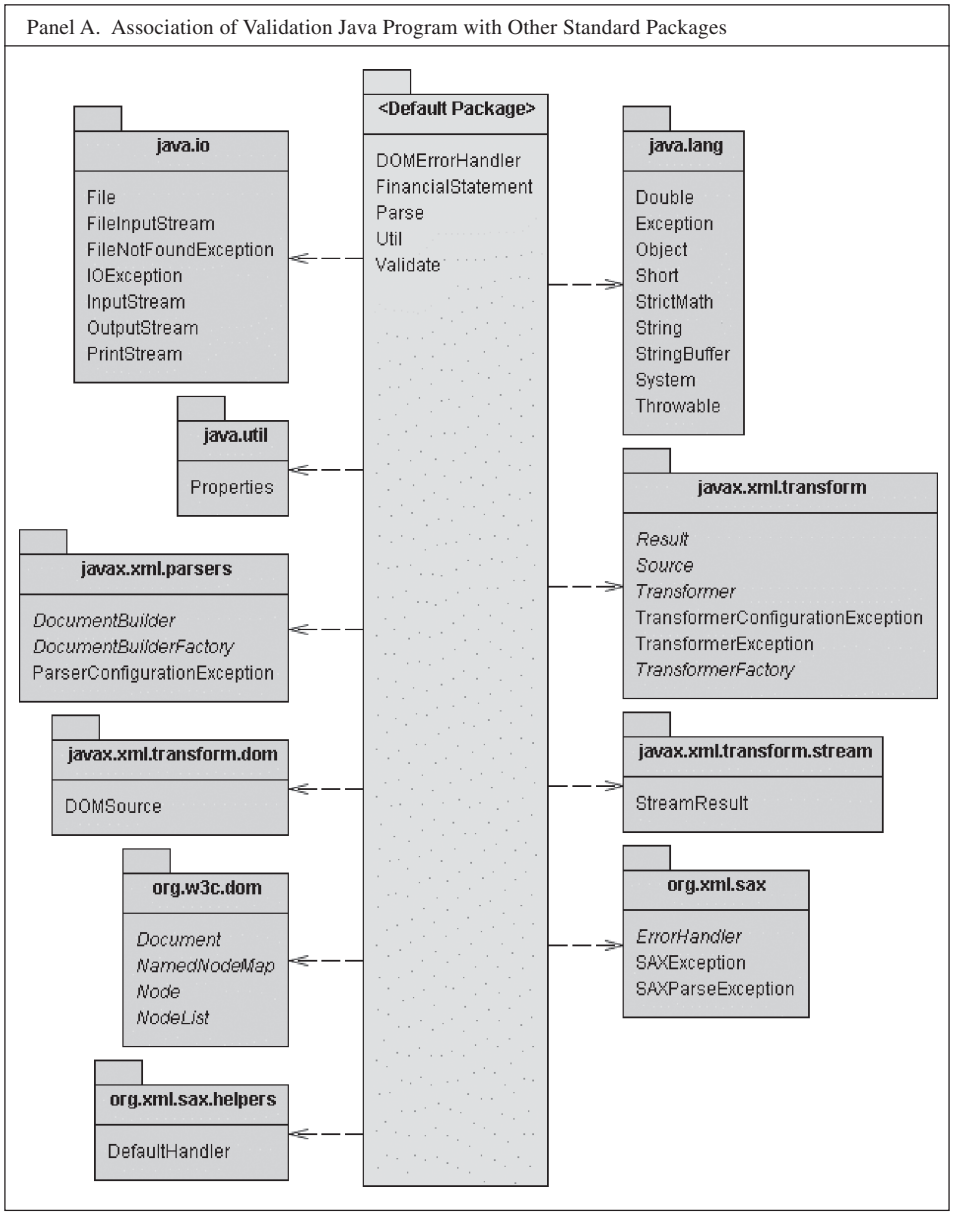
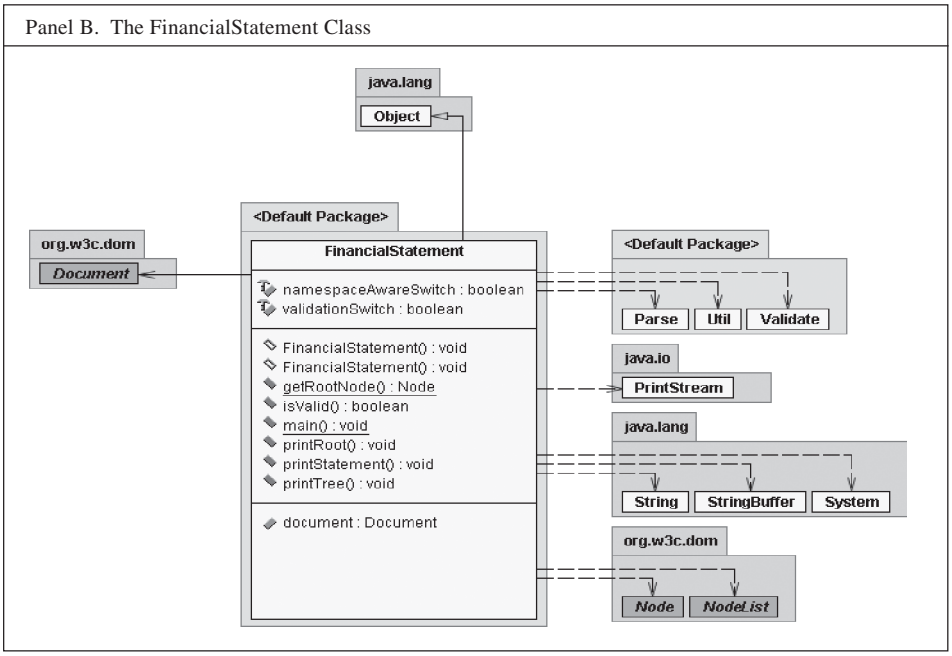


Figure 4. Validation Java Program in UML



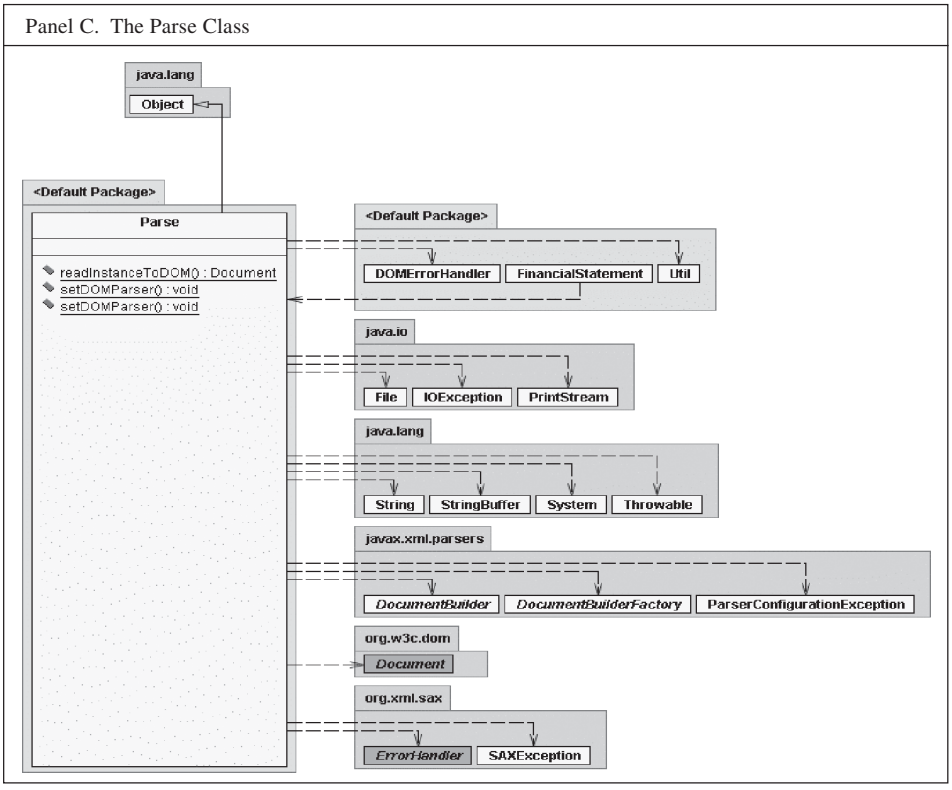
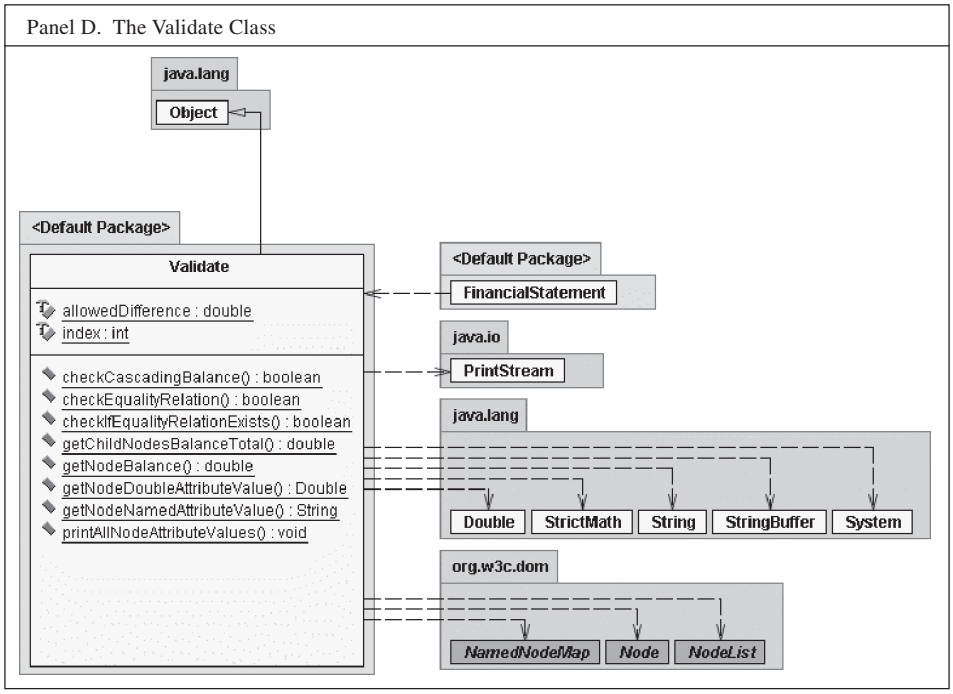


Figure 4 (Continued). Validation Java Program in UML



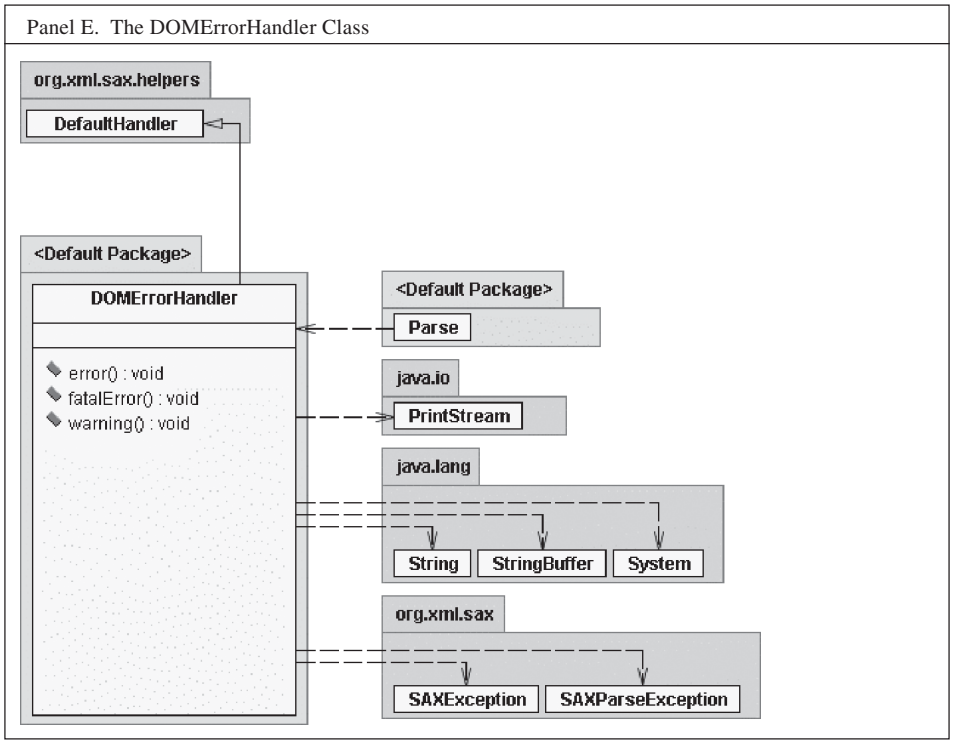


Figure 4 (Continued). Validation Java Program in UML

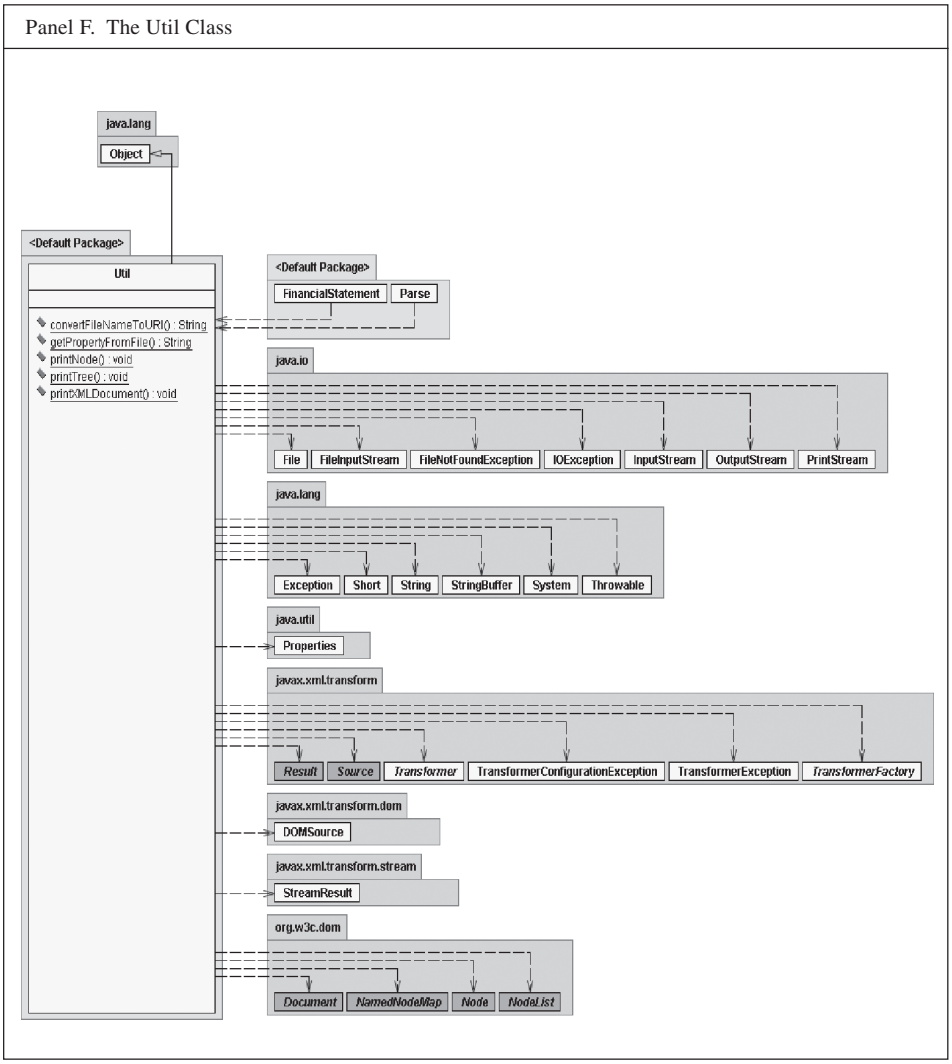


Figure 4 (Continued). Validation Java Program in UML

Illustration 1 presents source codes of three key methods for validating any financial statement against its schema. The «`getNodeNamedAttributeValue()`» method retrieves from a given node the value of a target attribute based on the attribute's name. This target attribute is then converted from the String type to Double type in the «`getNodeDoubleAttributeValue()`» method. The «`getNodeDoubleAttributeValue()`» method is used to retrieve values of the «`balance`» and «`relation`» attributes of a given node.

```
public static String getNodeDoubleAttributeValue (Node node, String attName)
{
NamedNodeMap nodeMap = node.getAttributes();
int length = nodeMap.getLength();
String value = null;
for (int i = 0; i < length; i++)
{
if ((nodeMap.item(i).getNodeName().trim()).compareTo(attName)==0)
{
value = nodeMap.item(i).getNodeValue();
break;
}
}
return value;
}

public static Double getNodeDoubleAttributeValue
(Node node, String attributeName)
{
String value = getNodeNamedAttributeValue(node, attributeName);
Double attributeValue = null;
if (value != null)
{
attributeValue = new Double(value);
}
return attributeValue;
}

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

Illustration 1. Excerpt of the JAVA-based Validation Program

The «`getChildNodesBalanceTotal()`» method scans each node for a list of all child elements. For each child element on the list, values of the «`balance`» and «`relation`» attributes are retrieved. The method sums up values of the balance attributes of all child elements. Depending on the value of the `relation` attribute, the contribution of each `balance` attribute to the accumulated sum could be positive or negative. This sum is to be compared, where applicable, with the parent's `balance` to check for compliance with financial statement rules.

The Java validation program, as illustrated by this excerpt, does not contain features specifically tied to any financial statement. This program is, therefore, generic to all financial statements. Results from the validation of various types of financial statements (see next subsection) verify its applicability to all financial statement instance documents.

5.3. A common directed tree model in XML for all financial statements

Illustrations 2 through 5 present XML schemas for the income statement, the cash flows statement, the statement of comprehensive income, and the statement of stockholders' equity. Each schema prescribes the directed tree structure of the associated financial statement. In designing financial statement schemas, this study adopts the terminology in the XBRL Commercial and Industrial Companies (C&I) taxonomy, but for the sake of simplicity in presentation, ignores some low level details (vun Kannon and Wang, 2000; XBRL, 2002).

Financial statements instance documents complying with their respective schemas are given in Illustrations 6, 8, 10, and 12. Finally, output from validating each instance document with respect to its corresponding XML schema is provided in Illustrations 7, 9, 11, and 13.

To demonstrate that the validating program can reject a violating financial statement, an income statement (Illustration 14) with an incorrect «`netIncome`» balance is used as input. The result of validation is given in Illustration 15.

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
<complexType name="netIncomeBeforeAcctingChangesType">
<all>
<element name="extraordinaryItems" minOccurs="0">
<complexType>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="netIncomeBeforeAcctingChanges"/>
</complexType>
</element>
<element name="netIncomeBeforeExtraordinaryItemsAcctingChanges"
minOccurs="0">
<complexType>
<all>
<element name="incomeFromDiscontinuedOperationsNetOfTax"
minOccurs="0">
<complexType>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="netIncomeBeforeExtraordinaryItemsAcctingChanges"/>
</complexType>
</element>
<element name="incomeFromContinuingOperations" minOccurs="0">
<complexType>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="netIncomeBeforeExtraordinaryItemsAcctingChanges"/>
</complexType>
</element>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="netIncomeBeforeAcctingChanges"/>
</complexType>
</element>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="netIncome"/>
</complexType>
<complexType name="netIncomeType">
<all>
<element name="cumEffectChangeAccountingPrinciple" minOccurs="0">
<complexType>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="netIncome"/>
</complexType>
</element>
<element name="netIncomeBeforeAcctingChanges"
type="xfs:netIncomeBeforeAcctingChangesType" minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="netIncomeAvailableToCommon"/>
</complexType>
```

Illustration 2. An XML Schema for Income Statement (is1.xsd)

```
<complexType name="netIncomeAvailableToCommonType">
<all>
<element name="preferredDividends" minOccurs="0">
<complexType>
<attribute name="relation" type="integer" fixed="-1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="netIncomeAvailableToCommon"/>
</complexType>
</element>
<element name="netIncome" type="xfs:netIncomeType" minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="incomeStatement"/>
</complexType>
<complexType name="incomeStatementType">
<all>
<element name="netIncomeAvailableToCommon"
type="xfs:netIncomeAvailableToCommonType" minOccurs="0"/>
<element name="earningsPerShareInformation" minOccurs="0">
<complexType>
<attribute name="relation" type="integer" fixed="1"/>
<attribute name="balance" type="decimal" fixed="0"/>
<attribute name="parent" type="NMTOKEN" fixed="incomeStatement"/>
</complexType>
</element>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="statement"/>
</complexType>
<element name="incomeStatement" type="xfs:incomeStatementType"/>
</schema>
```

Illustration 2. (Continued) An XML Schema for Income Statement (is1.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
<complexType name="depreciationAmortizationCashFlowsReconciliationType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="adjustmentsToReconcileCashFlows"/>
</complexType>
<complexType name="assetImpairmentChargeType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="adjustmentsToReconcileCashFlows"/>
.....
<complexType name="adjustmentsToReconcileCashFlowsType">
<all>
<element name="depreciationAmortizationCashFlowsReconciliation"
type="xfs:depreciationAmortizationCashFlowsReconciliationType"
minOccurs="0"/>
<element name="assetImpairmentCharge" type="xfs:assetImpairmentChargeType"
minOccurs="0"/>
```

Illustration 3. An XML Schema for Cash Flows Statement (cf1.xsd)
Note: Ellipses — sequence of dots (.....) is used to improve readability

```
.....
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="netCashFlowsOperatingActivitiesIndirect"/>
</complexType>
<complexType name="netIncomeType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="netCashFlowsOperatingActivitiesIndirect"/>
</complexType>
<complexType name="netCashFlowsOperatingActivitiesIndirectType">
<all>
<element name="netIncome" type="xfs:netIncomeType" minOccurs="0"/>
<element name="adjustmentsToReconcileCashFlows"
type="xfs:adjustmentsToReconcileCashFlowsType" minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="netCashFlowsOperatingActivities"/>
</complexType>
<complexType name="netCashFlowsOperatingActivitiesDirectType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="netCashFlowsOperatingActivities"/>
</complexType>
<complexType name="netCashFlowsOperatingActivitiesType">
<all>
<element name="netCashFlowsOperatingActivitiesIndirect"
type="xfs:netCashFlowsOperatingActivitiesIndirectType" minOccurs="0"/>
<element name="netCashFlowsOperatingActivitiesDirect"
type="xfs:netCashFlowsOperatingActivitiesDirectType" minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="netCashFlows"/>
</complexType>
<complexType name="netCashFlowsInvestingActivitiesType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="netCashFlows"/>
</complexType>
<complexType name="netCashFlowsFinancingActivitiesType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="netCashFlows"/>
</complexType>
.....
<complexType name="beginningOfPeriodCashAndCashEquivalentsType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="endOfPeriodCashAndCashEquivalents"/>
</complexType>
<complexType name="netCashFlowsType">
<all>
<element name="netCashFlowsOperatingActivities"
type="xfs:netCashFlowsOperatingActivitiesType" minOccurs="0"/>
<element name="netCashFlowsInvestingActivities"
type="xfs:netCashFlowsInvestingActivitiesType" minOccurs="0"/>
```

Illustration 3 (Continued). An XML Schema for Cash Flows Statement (cf1.xsd)
Note: Ellipses — sequence of dots (.....) is used to improve readability

```
<element name="netCashFlowsFinancingActivities"
type="xfs:netCashFlowsFinancingActivitiesType" minOccurs="0"/>
.....
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="endOfPeriodCashAndCashEquivalents"/>
</complexType>
<complexType name="endOfPeriodCashAndCashEquivalentsType">
<all>
<element name="beginningOfPeriodCashAndCashEquivalents"
type="xfs:beginningOfPeriodCashAndCashEquivalentsType" minOccurs="0"/>
<element name="netCashFlows" type="xfs:netCashFlowsType" minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="cashFlowsStatement"/>
</complexType>
<complexType name="supplementalDisclosureType">
<attribute name="relation" type="integer" fixed="1"/>
<attribute name="balance" type="decimal" fixed="0"/>
<attribute name="parent" type="NMTOKEN" fixed="cashFlowsStatement"/>
</complexType>
<complexType name="cashFlowsStatementType">
<all>
<element name="endOfPeriodCashAndCashEquivalents"
type="xfs:endOfPeriodCashAndCashEquivalentsType" minOccurs="0"/>
<element name="supplementalDisclosure"
type="xfs:supplementalDisclosureType" minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="statements"/>
</complexType>
<element name="cashFlowsStatement" type="xfs:cashFlowsStatementType"/>
</schema>
```

Illustration 3 (Continued). An XML Schema for Cash Flows Statement (cf1.xsd)
Note: Ellipses — sequence of dots (.....) is used to improve readability

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
<complexType name="foriegnCurrencyTranslationAdjustmentsType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="otherComprehensiveIncome"/>
</complexType>
<complexType name="changeInMinimumPensionLiabilityType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="otherComprehensiveIncome"/>
</complexType>
<complexType name="unrealizedGainsLossesType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="otherComprehensiveIncome"/>
</complexType>
<complexType name="reclassificationAdjustmentsType">
```

Illustration 4. An XML Schema for Statement of Comprehensive Income (ci1.xsd)
Note: Ellipses — sequence of dots (.....) is used to improve readability

```
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="otherComprehensiveIncome"/>
</complexType>
<complexType name="otherComprehensiveIncomeType">
<all>
<element name="foriegnCurrencyTranslationAdjustments"
type="xfs:foriegnCurrencyTranslationAdjustmentsType" minOccurs="0"/>
<element name="changeInMinimumPensionLiability"
type="xfs:changeInMinimumPensionLiabilityType" minOccurs="0"/>
<element name="unrealizedGainsLosses" type="xfs:unrealizedGainsLossesType"
minOccurs="0"/>
<element name="reclassificationAdjustments"
type="xfs:reclassificationAdjustmentsType"
minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="comprehensiveIncome"/>
</complexType>
<complexType name="netIncomeType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="comprehensiveIncome"/>
</complexType>
<complexType name="comprehensiveIncomeType">
<all>
<element name="netIncome" type="xfs:netIncomeType" minOccurs="0"/>
<element name="otherComprehensiveIncome"
type="xfs:otherComprehensiveIncomeType"
minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="statementOfComprehensiveIncome"/>
</complexType>
<complexType name="statementOfComprehensiveIncomeType">
<all>
<element name="comprehensiveIncome" type="xfs:comprehensiveIncomeType"
minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="statements"/>
</complexType>
<element name="statementOfComprehensiveIncome"
type="xfs:statementOfComprehensiveIncomeType"/>
</schema>
```

Illustration 4 (Continued). An XML Schema for Statement of Comprehensive Income (ci1.xsd)
Note: Ellipses — sequence of dots (.....) is used to improve readability

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.albany.edu/acc/xfs"
xmlns:xfs="http://www.albany.edu/acc/xfs"
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<include schemaLocation="bsj4.xsd"/>
<complexType name="balancePreferredStockType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="preferredStock"/>
</complexType>
```

Illustration 5. An XML Schema for Statement of Stockholders' Equity (se1.xsd)

```
<complexType name="changesPreferredStockType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="preferredStock"/>
</complexType>
<complexType name="balanceCommonStockType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="commonStock"/>
</complexType>
<complexType name="changesCommonStockType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="commonStock"/>
</complexType>
<complexType name="balanceRetainedEarningsType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="retainedEarnings"/>
</complexType>
<complexType name="changesRetainedEarningsType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="retainedEarnings"/>
</complexType>
<complexType name="preferredStockSharesType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="shares"/>
</complexType>
<complexType name="commonStockSharesType">
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="shares"/>
</complexType>
.....
<complexType name="preferredStockType">
<all>
<element name="balancePreferredStock" type="xfs:balancePreferredStockType"
minOccurs="0"/>
<element name="changesPreferredStock" type="xfs:changesPreferredStockType"
minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="balanceOfStockholdersEquity"/>
</complexType>
<complexType name="commonStockType">
<all>
<element name="balanceCommonStock" type="xfs:balanceCommonStockType"
minOccurs="0"/>
<element name="changesCommonStock" type="xfs:changesCommonStockType"
minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="balanceOfStockholdersEquity"/>
</complexType>
.....
<complexType name="retainedEarningsType">
<all>
<element name="balanceRetainedEarnings"
type="xfs:balanceRetainedEarningsType" minOccurs="0"/>
<element name="changesRetainedEarnings"
type="xfs:changesRetainedEarningsType" minOccurs="0"/>
```

Illustration 5 (Continued). An XML Schema for Statement of Stockholders' Equity (se1.xsd)

```
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="balanceOfStockholdersEquity"/>
</complexType>
.....
<complexType name="sharesType">
<all>
<element name="preferredStockShares" type="xfs:preferredStockSharesType"
minOccurs="0"/>
<element name="commonStockShares" type="xfs:commonStockSharesType"
minOccurs="0"/>
.....
</all>
<attribute name="relation" type="integer" fixed="0"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="statementOfStockholdersEquity"/>
</complexType>
<complexType name="balanceOfStockholdersEquityType">
<all>
<element name="preferredStock" type="xfs:preferredStockType"
minOccurs="0"/>
<element name="commonStock" type="xfs:commonStockType" minOccurs="0"/>
<element name="retainedEarnings" type="xfs:retainedEarningsType"
minOccurs="0"/>
.....
</all>
<attribute name="relation" type="integer" fixed="+1"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN"
fixed="statementOfStockholdersEquity"/>
</complexType>
<complexType name="statementOfStockholdersEquityType">
<all>
<element name="balanceOfStockholdersEquity"
type="xfs:balanceOfStockholdersEquityType" minOccurs="0"/>
<element name="shares" type="xfs:sharesType" minOccurs="0"/>
</all>
<attribute name="relation" type="integer" fixed="0"/>
<attribute name="balance" type="decimal" use="required"/>
<attribute name="parent" type="NMTOKEN" fixed="statements"/>
</complexType>
<element name="statementOfStockholdersEquity"
type="xfs:statementOfStockholdersEquityType"/>
</schema>
```

Illustration 5 (Continued). An XML Schema for Statement of Stockholders' Equity (se1.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="is0.xsl"?>
<incomeStatement balance="100" xmlns="http://www.albany.edu/acc/xfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.albany.edu/acc/xfs is1.xsd">
<netIncomeAvailableToCommon balance="100">
<netIncome balance="200">
<netIncomeBeforeAcctingChanges balance="700">
<netIncomeBeforeExtraordinaryItemsAcctingChanges balance="300">
<incomeFromContinuingOperations balance="100"/>
<incomeFromDiscontinuedOperationsNetOfTax balance="200"/>
</netIncomeBeforeExtraordinaryItemsAcctingChanges>
<extraordinaryItems balance="400"/>
</netIncomeBeforeAcctingChanges>
```

Illustration 6. An XML Income Statement instance (ISamd.xml)

```
<cumEffectChangeAccountingPrinciple balance="-500"/>
</netIncome>
<preferredDividends balance="100"/>
</netIncomeAvailableToCommon>
<earningsPerShareInformation/>
</incomeStatement>
```

Illustration 6 (Continued). An XML Income Statement instance (ISamd.xml)

```
The document file:C:\NOxfs\java\..\is2001Jun4\ISamd.xml is valid.
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation=http://www.albany.edu/acc/xfs is1.xsd
```

Illustration 7. Output from Validating the Income Statement

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="is0.xsl"?>
<cashFlowsStatement balance="1500" xmlns="http://www.albany.edu/acc/xfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.albany.edu/acc/xfs cf1.xsd">
<endOfPeriodCashAndCashEquivalents balance="1500">
<beginningOfPeriodCashAndCashEquivalents balance="1200"/>
<netCashFlows balance="300">
<netCashFlowsOperatingActivities balance="300">
<netCashFlowsOperatingActivitiesIndirect balance="300">
<netIncome balance="100"/>
<adjustmentsToReconcileCashFlows balance="200">
<minorityInterest balance="50"/>
<depreciationAmortizationCashFlowsReconciliation balance="10"/>
<writeOffAcquiredInprocessResearchAndDevelopment balance="20"/>
<assetImpairmentCharge balance="30"/>
<realizedGainsLossesOnSaleOfInvestments balance="90"/>
</adjustmentsToReconcileCashFlows>
</netCashFlowsOperatingActivitiesIndirect>
</netCashFlowsOperatingActivities>
<netCashFlowsInvestingActivities balance="500"/>
<netCashFlowsFinancingActivities balance="-500"/>
<effectOfExchangeRateOnCash balance="0"/>
</netCashFlows>
</endOfPeriodCashAndCashEquivalents>
<supplementalDisclosure/>
</cashFlowsStatement>
```

Illustration 8. An XML Cash Flows Statement instance (CFamd.xml)

```
The document file:C:\NOxfs\java\..\cf2001Sep25\CFamd.xml is valid.
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation=http://www.albany.edu/acc/xfs cf1.xsd
```

Illustration 9. Output from Validating the Cash Flows Statement

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="is0.xsl"?>
<statementOfComprehensiveIncome balance="1500"
xmlns="http://www.albany.edu/acc/xfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.albany.edu/acc/xfs ci1.xsd">
<comprehensiveIncome balance="1500">
<netIncome balance="1200"/>
<otherComprehensiveIncome balance="300">
```

Illustration 10. An XML Statement of Comprehensive Income instance (CIamd.xml)

```
<foriegnCurrencyTranslationAdjustments balance="300"/>
<changeInMinimumPensionLiability balance="500"/>
<unrealizedGainsLosses balance="-500"/>
<reclassificationAdjustments balance="0"/>
</otherComprehensiveIncome>
</comprehensiveIncome>
</statementOfComprehensiveIncome>
```

Illustration 10 (Continued). An XML Statement of Comprehensive Income instance (CIamd.xml)

```
The document file:C:\NOxfs\java\..\ci2001Sep25\CIamd.xml is valid.
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation=http://www.albany.edu/acc/xfs ci1.xsd
```

Illustration 11. Output from Validating the Statement of Comprehensive Income

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="is0.xsl"?>
<statementOfStockholdersEquity balance="1100"
xmlns="http://www.albany.edu/acc/xfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.albany.edu/acc/xfs se1.xsd">
<balanceOfStockholdersEquity balance="1100">
<preferredStock balance="300">
<balancePreferredStock balance="100"/>
<changesPreferredStock balance="200"/>
</preferredStock>
<commonStock balance="700">
<balanceCommonStock balance="300"/>
<changesCommonStock balance="400"/>
</commonStock>
<additionalPaidInCapital balance="30">
<balanceAdditionalPaidInCapital balance="5"/>
<changesAdditionalPaidInCapital balance="25"/>
</additionalPaidInCapital>
<retainedEarnings balance="30">
<balanceRetainedEarnings balance="15"/>
<changesRetainedEarnings balance="15"/>
</retainedEarnings>
<accumulatedOtherComprehensiveIncome balance="40">
<balanceAccumulatedOtherComprehensiveIncome balance="20"/>
<changesAccumOtherComprehensiveIncome balance="20"/>
</accumulatedOtherComprehensiveIncome>
</balanceOfStockholdersEquity>
<shares balance="1500">
<preferredStockShares balance="1200"/>
<commonStockShares balance="300"/>
</shares>
</statementOfStockholdersEquity>
```

Illustration 12. An XML Statement of Stockholders' Equity instance (SEamd.xml)

```
The document file:C:\NOxfs\java\..\se2001Sep25\SEamd.xml is valid.
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation=http://www.albany.edu/acc/xfs se1.xsd
```

Illustration 13. Output from Validating the Statement of Stockholders' Equity

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="is0.xsl"?>
<incomeStatement balance="100" xmlns="http://www.albany.edu/acc/xfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.albany.edu/acc/xfs is1.xsd">
<netIncomeAvailableToCommon balance="100">
<netIncome balance="500">
<netIncomeBeforeAcctingChanges balance="700">
<netIncomeBeforeExtraordinaryItemsAcctingChanges balance="300">
<incomeFromContinuingOperations balance="100"/>
<incomeFromDiscontinuedOperationsNetOfTax balance="200"/>
</netIncomeBeforeExtraordinaryItemsAcctingChanges>
<extraordinaryItems balance="400"/>
</netIncomeBeforeAcctingChanges>
<cumEffectChangeAccountingPrinciple balance="-500"/>
</netIncome>
<preferredDividends balance="100"/>
</netIncomeAvailableToCommon>
<earningsPerShareInformation/>
</incomeStatement>
```

Note: The Income Statement instance is arithmetically invalid because netIncome is not equal to `netIncomeBeforeAcctingChanges plus cumEffectChangeAccountingPrinciple.`

Illustration 14. An arithmetically invalid Income Statement instance (ISamd2.xml)

```
The document file:C:\NOxfs\java\..\is2001Jun4\ISamd2.xml is not valid.
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation=http://www.albany.edu/acc/xfs is1.xsd
```

Illustration 15. Output from Validating an Incorrect Income Statement

5.4. Summary observations

Using the balance sheet as illustration, Tam *et al.* (2002) propose a design of business reporting markup language based on a directed tree model. This study generalizes their findings by extending the directed tree model to the income statement, the cash flows statement, the statement of comprehensive income, and the statement of stockholders' equity. Results suggest that all financial statements can be expressed as a directed tree. The directed tree structure of any financial statement can be easily translated into XML. Because XML instance documents of all financial statements share the same directed tree structure, a single tree-traversal algorithm suffices for validating all statements.

The design of XML schemas as shown in Illustrations 2 through 5 adheres to desirable design criteria proposed by Tam *et al*. (2002)[9]. In particular, a standard

[9] Other desirable attributes may exist. Bovee *et al.* (2002), for instance, consider how well a taxonomy for financial statements corresponds to firms' preferred reporting practices.

modeling method (i.e. UML) is used in modeling. Resulting XML instance documents are very readable. XML schemas are modular, easy to maintain, and extensible through redefining types. Payload overheads for structural information are minimized. Now, this study demonstrates that a common validating routine exists for all financial statements expressed as directed trees. Therefore, Tam *et al*.'s (2002) XML schema design framework based on a common directed tree model actually conforms to all desirable design criteria.

6. CONCLUSIONS

All financial statements share a common directed tree model. This common structure, if appropriately taken into account in the design of markup languages, enables a generic validation routine to traverse and validate any financial statement expressed in XML as a directed tree. This brings significant savings in development and maintenance because separate validation routines for different financial statements become unnecessary.

XBRL is designed to facilitate electronic data exchange and analysis of financial statements. Because data in financial statement instances may be used for investment decisions, validation of electronically transmitted instance documents is imperative. This study exploits the common directed tree model of financial statements in the design of schema to standardize and simplify validation.

Taking advantage of the common directed tree structure in schema design enables instance documents of any financial statement to be validated by a single program. In order to establish the feasibility of this design approach, multiple schemas and instance documents are created. These schemas are based on the common directed tree structure of the income statement, the statement of cash flows, the statement of comprehensive income, and the statement of stockholders' equity. In addition, a single program for validating these instance documents against their schemas is also developed.

This study shows that a single generic validation program is sufficient to validate all instance documents of all financial statements against their schemas. This result contributes to a more efficient framework of XML taxonomy development for other industries. Because many other non-financial business documents are essentially hierarchical in structure, results from this study can be generalized to guide taxonomy development in other business areas.

7. REFERENCES

BONSÓN, E. (2001): "The Role of XBRL in Europe", *International Journal of Digital Accounting Research*, vol. 1, n. 2.

BOVEE, M.; ETTREDGE, M.; SRIVASTAVA, R.; VASARHELYI, M. (2002): "Does the Year 2000 XBRL Taxonomy Accommodate Current Business Financial Reporting Practice?", *Journal of Information Systems*, vol. 16, n. 2.

FpML. FpML: Call for Proposals. http://www.fpml.org/proposals/index.asp (visited July 18, 2002)

GENERAL ACCOUNTING OFFICE (GAO). (2002): Electronic Government: Challenges to Effective Adoption of the Extensible Markup Language. http://www.gao.gov/new.items/d02327.pdf

GIGA INFORMATION GROUP. (2001): Giga Survey: XML Achieving Mainstream Usage.

HOFFMAN, C.; KURT, C.; KORETO, R. (1999): "The XML Files", *Journal of Accountancy*, vol. 187, n. 5.

LUEDER, C. (2000): Issues Raised in DFAS XBRL Proof of Concept. http://xml.gov/documents/completed/dfas_xbrl_issues.htm

SUN MICROSYSTEMS. (1994): XML AT SUN FAQs. http://www.sun.com/software/xml/faqs.html#3

TAM, K.; GOEL, S.; GANGOLLY, J. (2002): "On the Design of an XML Schema Based Application for Business Reporting: An XBRL Schema Perspective", *International Journal of Digital Accounting Research*, vol. 2, n. 1.

VASAL, V.; SRIVASTAVA, R. (2002): "eXtensible Business Reporting Language (XBRL) – The Digital Language of Business: An Indian Perspective", *Indian Accounting Review*, vol. 6, n. 1.

VUN KANNON, D.; WANG, Y. (2000): Design of the XBRL specification. http://www.infoloom.com/gcaconfs/WEB/paris2000/S26-01.HTM (visited June 6, 2001)

W3C (2000): Extensible Markup Language (XML) 1.0 (Second Edition). http://www.w3.org/TR/2000/REC-xml-20001006 (visited May 15, 2002)

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

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

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

XBRL (2000): Extensible Business Reporting Language (XBRL) Specification. http://www.xbrl.org/resourcecenter/specifications.asp (visited November 8, 2003)

XBRL (2001): Extensible Business Reporting Language (XBRL) 2.0 Specification. http://www.xbrl.org/resourcecenter/specifications.asp (visited November 8, 2003)

XBRL (2002): US GAAP Commercial and Industrial (Presentation Report). http://www.xbrl.org/taxonomy/us/fr/gaap/ci/2002-10-15/us-gaap-ci-2002-10-15-elements.pdf.