LIGHTWEIGHT APPROACH OF XML IMPLEMENTATION AS INDUSTRIAL STANDARD FOR INFORMATION SHARING

ERWIN WIDODO
Industrial Engineering Department, Sepuluh Nopember Institute of Technology, ITS, Surabaya.
Kampus ITS Sukolilo Surabaya 60111
erwin@ie.its.ac.id

ABSTRACT

Information sharing in supply chain has emerged as one critical factor in maintaining company's responssiveness. This ability is needed to overcome both the supplier and customer requirements in sharing their data to accomplish their operational routines. A lot of efforts have been devoted in such research area including XML (eXtensible Markup Language) technology. However, a stepwise approach on how to utilize this advance technology is still lacking. This paper proposed a lightweight approach to implement XML in information sharing firmly. The main XML characteristics will be exploited so that the advantages of utilizing this mark up language may be highlighted. Based on these important remarks a lightweight approach of implementing XML is possible to be composed. The expected outcome is a step-by-step guidance for data exchanger to utilize XML simply but precisely and accurately. A case study of information sharing in commerce is elaborated to enhance our understanding on how to implement this approach in real domain problem.

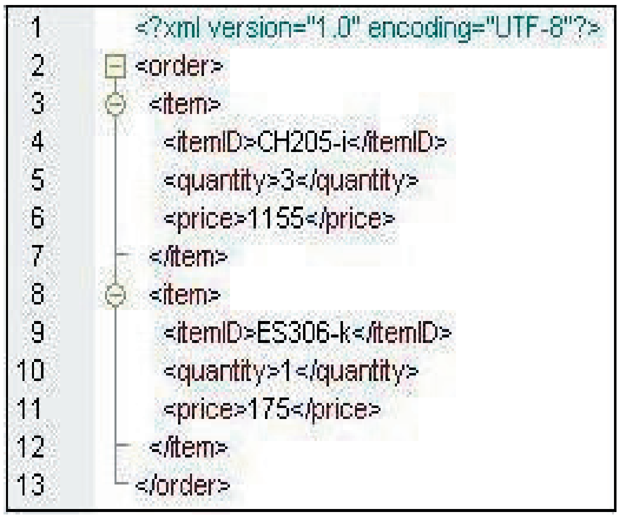
***Key words**: XML, information sharing*

INTRODUCTION

The usage of information in companies' decision making is definitely important in current global environment. The demand of fast, interchangeable, and reliable information is rapidly increasing. From supply chain point of view, a company needs to have an access to its customer demand information in preparing its own production or service output. On the other hand, it is also important for this company to discern its supplier information in providing the required inputs. XML (eXtensible Markup Language), one type of markup language, becomes one feasible solution to answer this challenge.

Basically XML is one subset of SGML (Standard General Markup Language). This markup language was proposed by W3C (World Wide Web Consortium) in February 1998. Its primary purpose is to facilitate the data sharing across different information systems, particularly among several systems connected via internet. XML provides a text-based mean to describe and apply a tree structure of represented information. At its base level, all information is manifested as text, equipped with markup that indicates the information's separation into a hierarchy of container-like elements altogether with attributes and character data of those elements. Figure 1 below shows a example of truncated XML document in commerce area. There is a set of order information which includes two (2) different items ordered accompanied with their details, namely item ID, quantity, and price.

Regarding its characteristics, there are a number of XML main advantages, such as: It is platform-independent, thus relatively immune to changes in technology; Its format which is based on international standards, human and machine readable; It has support for Unicode, allowing almost any information to be communicated; It has ability to represent the most general computer science data structures: records, lists and trees; User may perform self-documenting format that describes user-own structure and field names; XML document can be parsed by using its structure to process the data.



```
<?xml version="1.0" encoding="UTF-8"?>
<order>
  <item>
    <itemID>CH205-i</itemID>
    <quantity>3</quantity>
    <price>1155</price>
  </item>
  <item>
    <itemID>ES306-k</itemID>
    <quantity>1</quantity>
    <price>175</price>
  </item>
</order>
```

Figure 1. Example of a simplified XML representation

Besides those strong points, this markup language is also preferable because of some supporting reasons such as plain text manifestation and extensive experience inherited from previous markup language. Having realized the merits of XML, data exchange operators have a very promising solution in overcoming the main hurdle in data exchange operations among supply chain echelons.

Most companies store their data and information in various format, including paper documents, electronic documents, spreadsheets, and relational databases This situation leads for interoperability problem when they want to manage their information sharing. Therefore, by utilizing a common information sharing format the individual industrial partners do not need to invest heavily in new and interoperable computer systems or databases. They just need to format information in a common structure. In this case, XML is definitely the proper solution. Moreover, this solution is also becoming easier as most used word processors and spreadsheet programs allow for the creation of XML documents.

The usage of XML will become more effective in cross-industrial border information sharing when a step wise approach is available. A lot of researches have been done about XML, nonetheless, there is still lack on providing a set of step by step operational guidance how to gain the merits of this beneficial document in information sharing. Based on this problem, this paper tries to provide a bridging result between XML strong points and the favorable XML operational routines by proposing a lightweight approach in XML implementation as an industrial standard for information sharing.

The rest of this paper is organized as follows: Figure 2 describes the XML characteristics by which become the underlying justification to compose the aimed approach. Figure 3 elaborates the main section of this paper, the lightweight approach. There exists some step-by-step guidance for information sharing routines. Afterwards, a case study of *XIS*, *XML-based Information Sharing* in commerce area is presented in figure 4. This section is prepared to enhance reader's understanding about the proposed approach. As usual, the next part, figure 5, sums up some important remarks of this research. Finally, a short reference of future work as a further expansion of this research ends this paper.

XML possesses some specific characteristics which makes this format is very useful in information sharing. In this section, there is a brief review of them in each associated sub-sections. Regardless the type of business, either B2B (business to business) or B2C (business to customer), these characteristics are very beneficial in performing information sharing among supply chain players.

Compare to HTML, XML provides a "user-defined tagging" system in marking up their data with appropriate tag characters. By using XML, information operators are free to determine the root element tag, element tag, sub-element tag, sub-sub-element tag, and so on. When the attributes are necessary, users also have freedom to concatenate those attributes tag with associated element tag (c.f. section, advantage of XML no. 5). Contrarily, when data operators have to deal with the previous version of markup language, HTML, they have to stay focus in creating, processing, displaying, and maintaining a mixture of data and its structure in a single document in addition to inflexible vocabulary of elements and attributes. The Basic syntax of XML is:

```
<name attribute="value">content</name>
```

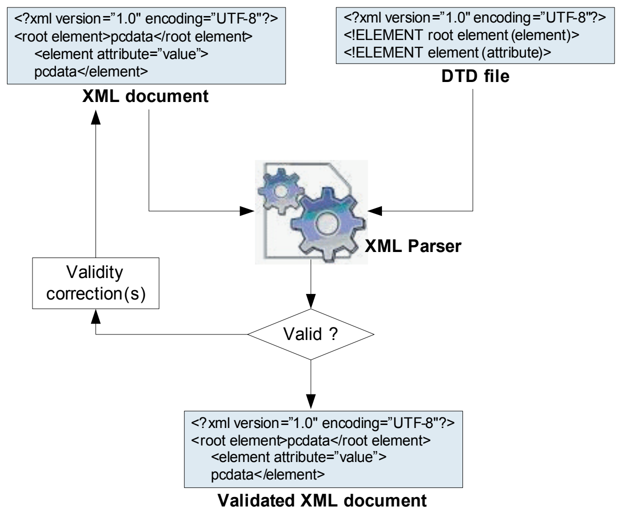
Name refers to the tag name for element under consideration. Tag name for one particular non-empty element always has start tag name surrounded by angle brackets (<name>) and end tag name also surrounded by angle brackets with slash preceding the name (</name>). Attribute is a specific value for an element included in the start tag. An attribute value should be quoted using single or double quotes. One attribute name only appears once in any elements. Content is the main data for element under consideration. Everything written between start and end tag is the content of element under consideration. Users are free to give names of these name and attributes, unlike in HTML which is very strict to the rules of SGML and EBNF (Extended Backus-Naur Form).

Even XML users have sufficient freedom to construct their documents, there are still two restraining rules to be followed. The first one is known as "well formedness." This rule is fulfilled when all of document contents conform to all of XML's syntax rules. A document that is not well-formed is not considered to be XML; a parser, an analyzer of XML document, is not allowed to process it. For example, if a non-empty element has an opening tag with no closing tag, this XML document is *not well-formed.*

Second rule in constructing XML documents is "validity." A valid XML document has data that conforms to a particular set of "user-defined content rules." These rules are commonly represented by Schemas. An XML Schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic constraints imposed by XML itself. A number of standard and proprietary XML schema languages have emerged for the purpose of formally expressing such Schemas, and some of these languages are XML-based, themselves.

For example, if an element in one particular branch of a document tree structure is required to contain text that can be interpreted as being an integer numeric value, and it instead has the text "hello", or has other elements, then the document is considered as *not valid*.

In performing such validity, user have to parse the input XML document altogether with the casting DTD file, the oldest Schema format for XML, into a specific validating system. The main component of this system is XML parser. There is a number of XML parser available freely, however, Xerces seems to be the most famous one. Figure 2 reflects how DTD works towards validating process.



<?xml version="1.0" encoding="UTF-8"?>
<root element>pcdata</root element>
<element attribute="value">
pcdata</element>
XML document
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT root element (element)>
<!ELEMENT element (attribute)>
DTD file
XML Parser
Validity correction(s)
Valid ?
<?xml version="1.0" encoding="UTF-8"?>
<root element>pcdata</root element>
<element attribute="value">
pcdata</element>
Validated XML document

Figure 2. XML validation process using DTD

XML is a data description language that is why XML documents do not carry information about how to display the data. Without using additional representation files such as CSS (Cascading Style Sheet) or XSLT (XML Style Sheet Language Transformation), a generic XML document is rendered as raw XML text by XML viewers (in most cases, these refer to web browsers). Some display it with 'handles' (e.g. + and - signs in the margin) that allow parts of the structure to be expanded or collapsed with mouse-clicks.

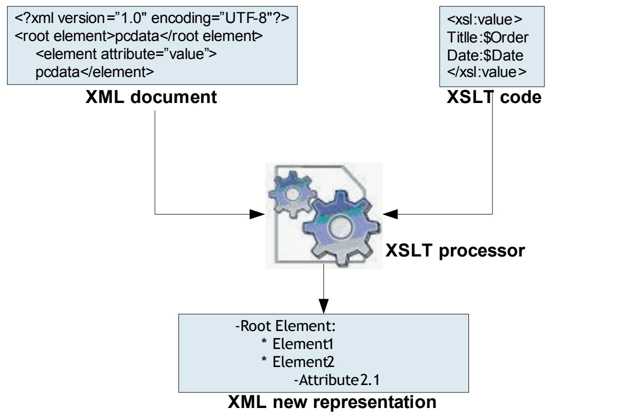


Figure 3. Relationship among XML and XSL

Extensible Stylesheet Language (XSL) can be used to alter the format of XML data, either into HTML or other formats that are suitable for a browser to display. In transforming the original XML document, an XSL Transformation (XSLT) file is required. Figure 3 shows how transformation of XML document into desirable style. Inputted XML document is processed together with its transformer XSLT code using associated processor. The result is the new representation of XML document.

METHODS

Ability to Process the Data. Information sharing operator may need to process the data in managing their tasks. Common processes to be undertaken are creating a new XML document, adding a record within an existing XML file, modifying a record, or deleting a record. These operations can be completed by performing data binding concept The basic idea of this concept is to utilize associated Java classes, since Java language is known as the XML's best counterpart. XML document can be used as a transfer mechanism between source database and software application. This transfer is performed by binding a Java object (a Java class instantiation) to an XML document. Data binding can be implemented by generating a Java class to represent the restrictions of DTD or other schemas. User may utilize this Java class to create a valid XML document based on its DTD or schema, to read the document, and to validate it as well.

There are still a lot of XML processes, especially employing advance technology, such as data binding based on schema in addition to DTD, using SAX, Simple API (Application Programming Interface) for XML, handler to parse XML document, or data transfer using XML database component. However, these topics will become the author's future works as continuation of current research presented in this paper.

The Proposed Lightweight Approach. Based on the XML characteristics aforementioned before, a stepwise lightweight approach in implementing XML document can be proposed as follows.

Initial XML File Creation. Prerequisite of creating initial XML document is to determine its basic tree structure. In every XML document there is. This initial step can be expanded as:

Establish the root element as basic, as main expansion point of following elements as data containers. There is always one root element in an XML document. Determine the child (children) element, as the offspring of the root element. Disentangle further grand-child (children) element, if available.

Assign attribute correctly besides the child element. In differentiating the usage of elements and attributes in branching the document's tree structure, users have to consider the variability of either elements and attributes value. In case the considered value has no variability, choosing element as the branching representation is the correct decision. Contrarily, when users find one branching value is various, attribute is the correct one to be chosen.

Represent the above design using XML development environment tool. This step is the important one since it brings to reality what kind of XML document is being created. There is a wide range of such application tools. Assuming that the operating system is under Windows, users may use tools starting from the simple one Notepad, Wordpad, to the famous word processor MS Word or even its sibling MS Excel for spreadsheet.

Well-formedness and Validity Assurance. Considering the time saving, XML users usually employ XML development environment tool to perform well-formedness and validity test. However, the basic testing steps are:

Checking whether all XML syntaxes are conformed to the SGML and EBNF rules. These rules address to well-formedness criterion. For important note, the big two (2) common mistakes fall in this category are: a) Neglect to pair the open tag with its close tag in the end of one branch. This mistake happens when users deal with a complex XML document but apply a simple development environment by which has no automatic tag pairing check; and b) Consistency of tag naming, especially for "case sensitive" rule. Users sometimes are inattentive to be consistent in writing small letters or capital ones.

Employing XML Schema (most used one is DTD) as the grammar to determine whether all elements and attributes within XML document have been placed on the correct nodes and carried the proper contains. This restriction is the manifestation of validity criterion.

Concise Representation. Basically there are two ways to represent XML documents:

The first way is using the XML plain text nature. Users may display their XML file by applying a number of XML development environment tools. There is a wide range of such application tools. Assuming that the operating system is under Windows, users may use tools starting from the simple one Notepad, Wordpad, to the famous word processor MS Word or even its sibling MS Excel for spreadsheet.

The second one is using a transformer file, either CSS or XSL file. There is a In order to style the rendering in a viewer with CSS, the XML document must include a reference to the stylesheet:

```
<?xml-stylesheet type="text/css"

href="theStylesheet.css"?>
```

When the user needs to specify client-side XSLT, the following processing instruction is required in the original XML document:

```
<?xml-stylesheet type="text/xsl"

href="theTransformer.xsl"?>
```

Note that this is different from specifying such a stylesheet in HTML, which uses the <link> element.

XML Processing. XML processing mainly deals with creating new XML documents, adding a new element (record) within existing XML document, as well as modifying it.

Creating a new XML document is a must before performing other actions. Please refer to sub-section 3.1 aforementioned.

Adding a new element, or can be referred as record, requires some special actions. Users have to compose an associated Java class to perform data binding. Within this Java class, an adding Java method must exist.

Resembling adding process, modifying an XML element also needs a Java class to bind the data. The emphasis on this action is unmarshalling (read) the original XML document first, modifying it, and then marshalling (write) the modified one.

Exemplary case study elaborated in next section will give clearer understanding on these adding and modifying XML element (record).

RESULT AND ANALYSIS

In this section, a case study of commerce domain problem is employed. Let us name the domain under discussion with XIS (XML-based Information Sharing) case study. In commerce domain, we have a lot of business processes which is suitable to be depicted as an exemplary case such as procurement, inventory, accounting, and so on. However, "ordering" process seems to be the most understandable and the simplest activities to be sampled. From now on, the case focus is only on information sharing of ordering activity.

Based on the proposed approach, the first thing to be done is to initiate the prime XML file. When users have to prepare an order file, at least there have to exist information about the ordered item, how many/much of quantity needed, and the associated price. Hence, users have to determine that: a) The root element is "order"; and b) The children elements are "itemID" for unique field of one particular item ordered, "quantity" for the amount of item needed, and "price" to record the nominal value of one single ordered item.1

Figure 4 gives a conceptual tree structure of XIS case study.

Considering the simplicity of XIS case to be understood easily and due to space limitation, grand children elements and attributes can be omitted. Figure 1 in section 1 is a sample XML of document.

Second step is to assure the well-formedness and validity. In case of well-formedness, the only way to get passed is to conform to all SGML rules for grammar rules despite of the freedom characteristics of XML in preferring its tagging names. User can take advantage by using XML editor in performing such action. There are some editors which is freely available despite some well-equipped ones are usually commercial products. As a recommendation, XML users may use XMLSpy from Altova to create and to edit XML document. In case of implementing joint operation between XML files and Java classes, JBuilder from Borland is considered as one of the most favorite integrated development environment. When non-commercial tools are preferred, Eclipse and Sun's NetBeans are favorable.

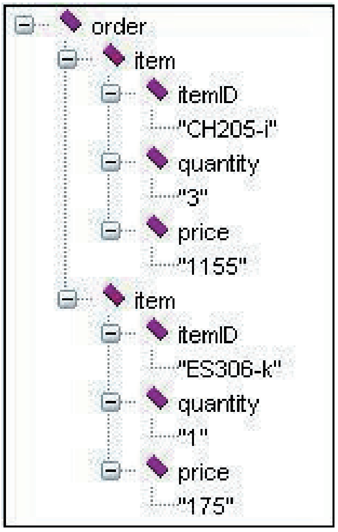
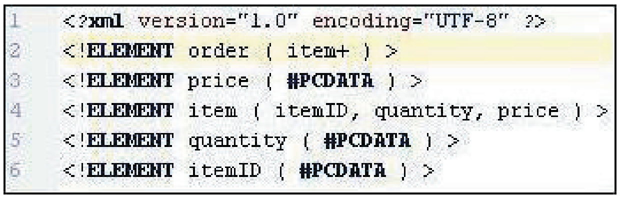


Figure 4. Tree structure of XIS



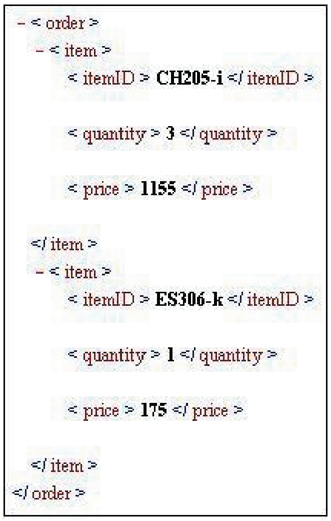
```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT order ( item+ ) >
<!ELEMENT price ( #PCDATA ) >
<!ELEMENT item ( itemID, quantity, price ) >
<!ELEMENT quantity ( #PCDATA ) >
<!ELEMENT itemID ( #PCDATA ) >
```

Figure 5. DTD for XIS

In performing validity check, a DTD file is needed. This DTD file is used to assure all elements and data within XML document are properly placed and contained. DTD for XIS is represented in figure 5.

Having got the ready to process XML file, the next step is to represent it concisely. In case the user requires no additional style, about the same display of original XML file will be represented as shown in figure 6.

On the other hand, when the viewer demands a modified appearance, a CSS (Cascading Style Sheet) is the first alternative. This CSS file can be used to assist the XML viewer application to show tidier representation by eliminating XML tags. Figure 7 is the depiction of representation result accompanied with corresponsding CSS file.



```
- < order >
  - < item >
      < itemID > CH205-i </ itemID >

      < quantity > 3 </ quantity >

      < price > 1155 </ price >

    </ item >
  - < item >
      < itemID > ES306-k </ itemID >

      < quantity > 1 </ quantity >

      < price > 175 </ price >

    </ item >
</ order >
```

Figure 6. The representation of XIS document without additional representation style

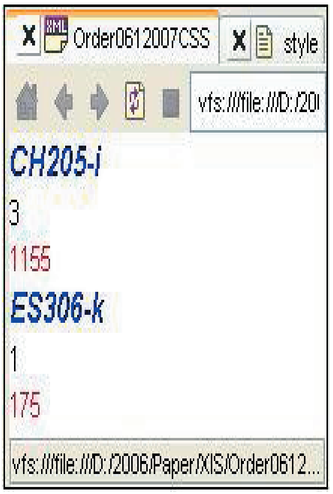
In case this CSS representation is still not enough, an XSLT representation employing XSL file provides better solution. Figure 8 shows the representation of XIS document empowered by its XSL on its right-hand side. The last step is to process the ready and well-displayed XIS document. Two (2) main actions to be performed here are:

Adding a new element (in database, it can be referred as record as well), namely Added Item to the current XIS document (exactly Order0612007Pro.xml).

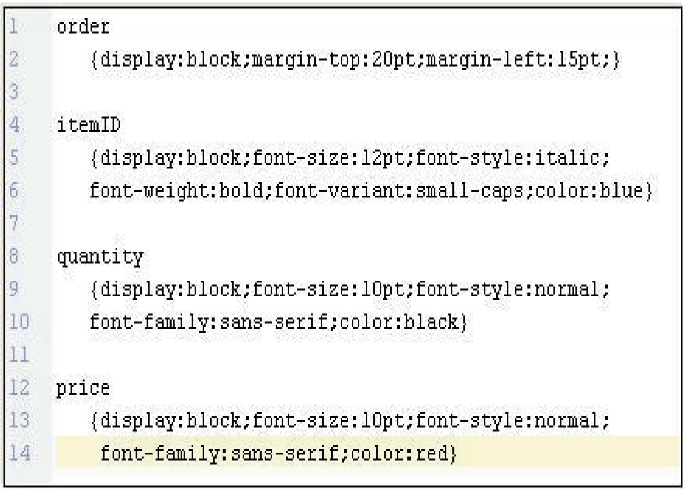
Modifying the last element, by mean to replace the last element of Order0612007Pro.xml file (ES306-k) with the new element specified by the input in Java code (Modified Item). Figure 9 shows the cutlet of Java code to perform such adding and modifying element actions of Order0612007.xml file.

When we run these Java class, the output resulted in Java IDE tool console is sbhown in the following figure 10. This figure shows that Order0612007Pro.xml has been parsed by XML parser, added and modified with the specified data by associated Java code depicted in figure 9.

In the first unmarshalling section, the original Order0612007Pro.xml was processed. The number



```
CH205-i
3
1155
ES306-k
1
175
```



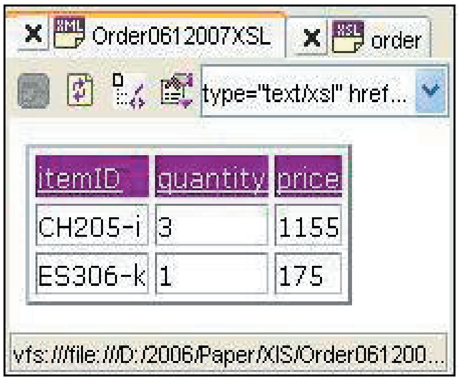
```
order
   {display:block;margin-top:20pt;margin-left:15pt;}

itemID
   {display:block;font-size:12pt;font-style:italic;
   font-weight:bold;font-variant:small-caps;color:blue}

quantity
   {display:block;font-size:10pt;font-style:normal;
   font-family:sans-serif;color:black}

price
   {display:block;font-size:10pt;font-style:normal;
    font-family:sans-serif;color:red}
```

Figure 7. The representation of XIS document using CSS file

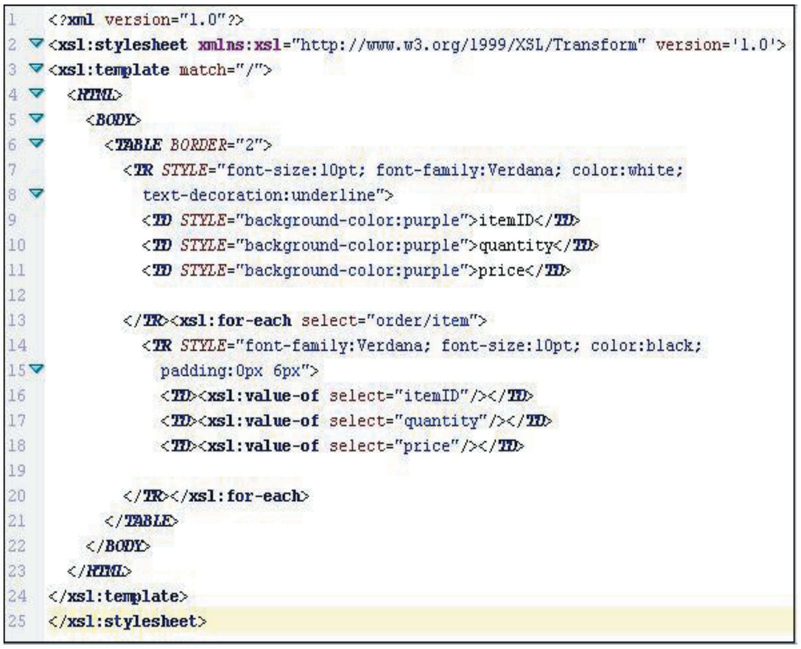
of items being read is two (2). There are also a couple of data about the first and last item within the original order document. This output shows that Java code for data binding functionality has been working well. Second section shows the adding process result. The number of items is three (3) not two anymore and the last ordered item is Added Item, not ES306-k anymore. This result indicates that the adding process is also performed well. The last part of that result mentions that the last item has been changed from Added Item to Modified Item. This shift points out that modification process also has taken place properly.

CONCLUSION

Based on the theoretical proposed approach and practical case study, we may conclude some important points as follows:

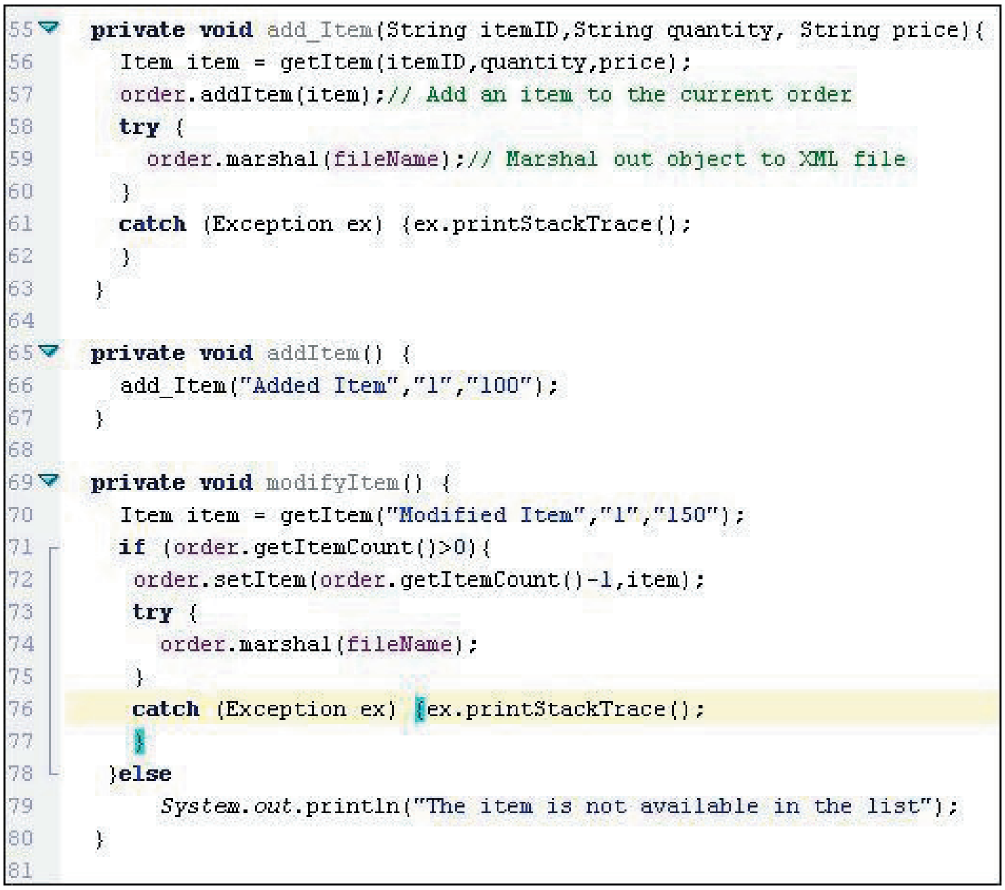


itemID	quantity	price
CH205-i	3	1155
ES306-k	1	175



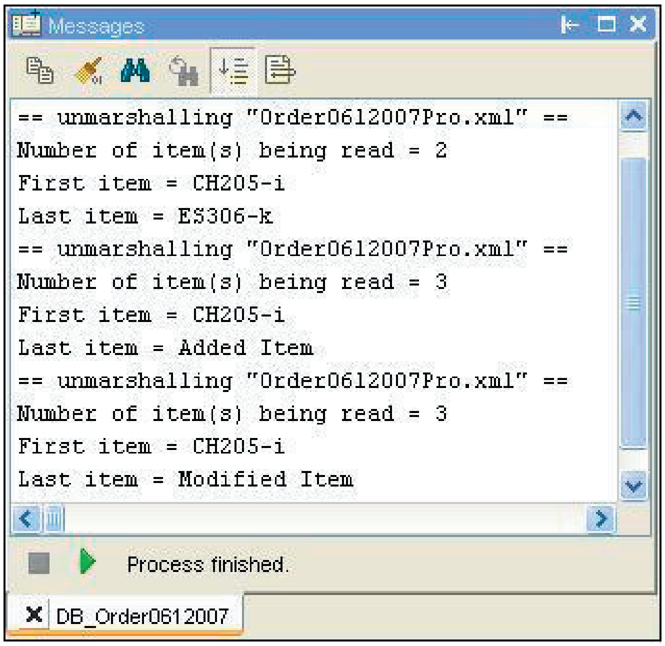
```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version='1.0'>
<xsl:template match="/">
  <HTML>
    <BODY>
      <TABLE BORDER="2">
        <TR STYLE="font-size:10pt; font-family:Verdana; color:white;
          text-decoration:underline">
          <TD STYLE="background-color:purple">itemID</TD>
          <TD STYLE="background-color:purple">quantity</TD>
          <TD STYLE="background-color:purple">price</TD>

        </TR><xsl:for-each select="order/item">
          <TR STYLE="font-family:Verdana; font-size:10pt; color:black;
            padding:0px 6px">
            <TD><xsl:value-of select="itemID"/></TD>
            <TD><xsl:value-of select="quantity"/></TD>
            <TD><xsl:value-of select="price"/></TD>

        </TR></xsl:for-each>
      </TABLE>
    </BODY>
  </HTML>
</xsl:template>
</xsl:stylesheet>
```

Figure 8. The representation of XIS document using XSL file



```java
  private void add_Item(String itemID,String quantity, String price){
    Item item = getItem(itemID,quantity,price);
    order.addItem(item);// Add an item to the current order
    try {
      order.marshal(fileName);// Marshal out object to XML file
    }
    catch (Exception ex) {ex.printStackTrace();
    }
  }

  private void addItem() {
    add_Item("Added Item","1","100");
  }

  private void modifyItem() {
    Item item = getItem("Modified Item","1","150");
    if (order.getItemCount()>0){
     order.setItem(order.getItemCount()-1,item);
     try {
       order.marshal(fileName);
     }
     catch (Exception ex) {ex.printStackTrace();
     }
   }else
       System.out.println("The item is not available in the list");
  }
```

Figure 9. Java code for XIS data binding



```
== unmarshalling "Order0612007Pro.xml" ==
Number of item(s) being read = 2
First item = CH205-i
Last item = ES306-k
== unmarshalling "Order0612007Pro.xml" ==
Number of item(s) being read = 3
First item = CH205-i
Last item = Added Item
== unmarshalling "Order0612007Pro.xml" ==
Number of item(s) being read = 3
First item = CH205-i
Last item = Modified Item
```

Figure 10. Console display of adding and modifying result

In this paper, lightweight approach to implement XML as industrial standard for information sharing is proposed. This approach provides a step-by-step basic guidance how to process XML as information exchange media. Applying this approach utilizing internet-based system, users gain the advantages of XML to make their task easier, to ensure the seamless of information sharing processes, and to enhance the validity of information exchange as well. The example explained in previous section shows that all task within the proposed approach can be smoothly applied in a commerce problem domain, XIS.

This proposed approach works well in conjunction with some additional Java classes. This programming language was chosen because of its characteristics, write once, read anywhere (WORA). This platform-independent and non-commercial technology helps the proposed approach in terms of automatically generating Data Type Document (DTD) required and performing data binding of XML processing. However, several users may prefer to use some commercial language programming regarding some special reasons. In such case, some modification may be required in performing the proposed approach.

Once this lightweight approach has been accepted by larger information-based industrial community, the development of a further set of XML implementation consensus can provide more beneficial contribution towards information sharing and E-Business among industry partners, non-profits organizations and public sector entities.

Future Work

To create a comprehensive representation of the proposed approach, the author has a plan to develop an advance supporting system of XML processing tool. This system will have ability to be a development environment tool for XML document creation, reviewing, well-formedness checking, and validation as the basic functionality. Moreover, this approach may generate multiplying effect when it is applied in industrial groups. Typically, XML standards are developed through industry collaboration which includes entities such as major industry participants, non-profit organizations or neutral web services firms. The purpose of these industry groups is focused on continuously improving XML implementation guidelines, standards, and terminology that can be applied throughout the industry, for example developing tagging dictionary for one particular group of industry (c.f. sub-section 2.1). This fact becomes the background of the author's next future work, to compose a framework of XML implementation within a certain industrial collaboration.

REFERENCES

Archiniegas, F. XML Developers Guide. Osbourne McGraw-Hill Publishing, 2001.

Bray, T. *A conversation with Tim Bray: Searching for ways to tame the world's vast stores of information*. Association for Computing Machinery's "Queue site", 2006.

Bray, T., Jean Paoli, C., Sperberg-McQueen M., Maler E., Yergeau F. *Extensible Markup Language (XML) 1.0 (Fourth Edition) - Origin and Goals*. World Wide Web Consortium. Retrieved on October 29, 2006.

Brown, D.A., Tittle, E. *Schaum's Easy Outline XML: Outline of Theory and Problem of XML*. McGraw-Hill Co., 2004.

Castro, E. *HTML for the World Wide Web: Visual QuickStart Guide, Fifth Edition with XHTML and CSS*. Peachpit Press, 1999.

Cate, F.H., Staten, M.E. *The Value of Information Sharing*. Protecting Privacy in the New Millennium Series, 2000.

DeRose, Steven J. *The SGML FAQ Book*. Boston: Kluwer Academic Publishers, 1997.

Harold, E.R., Means, W.S. *XML in Nutshell (third edition)*. O'Reilly Media Inc., CA, 2004.

Horton, I. *Ivor Horton's Beginning Java 2 ™ SDK 1.4 Edition*. Wiley Publishing Inc., IN, 2003

Main page for World Wide Web Consortium (W3C) XML activity and information. http://www.w3.org/XML.

McLaughin, B., Loukides, *M. Java and XML*. O'Reilly Media Inc., CA, 2000.

Vlist, E. *Relax NG*. O'Reilly Publisher, 2003.

Why XML, Software AG, http://www.epa.gov/ epaoswer/osw/conserve/plugin/.