Running Head: The Case Against XML

The Case Against XML

Kristofer J Carlson

Bowie State University

Abstract

XML is a highly touted technology. While originally intended as a presentation format for the web, it has morphed into a technology concerned with data. Proponents of XML have adopted the hierarchal tree structure as the data format for XML. This creates a great many real world difficulties, as many problems cannot be adequately represented through the use of tree structures. XML encapsulates the data with tags, and the resulting encapsulated data can be widely distributed and then recreated as necessary. This is perhaps where the idea of the distributed XML database comes from. XML is also widely discussed as a data exchange format. XML is readable and self-describing, but is also extremely verbose. Since the meaning of the data has been previously agreed upon, the tags are redundant. XML is likely a fad technology which will flourish until its problems become evident, at which time it will dwindle into a niche product.

The Case Against XML

XML, or Extensible Markup Language, is generally touted as a sort of cure-all for all manner of ills. For example, ZapThink, LLC. says "eXtensible Markup Language (XML) will soon be everywhere. It will run your business, your industry, and our world (ZapThink, 2001, p. 47). Through the universal use of XML, we are to be ushered into a brand new era where everything can talk to everything else, where data can be freely shared between multiple programs with vastly different data formats, and where document validity is assured. While the promise of XML is often touted, and while an elementary exposure to XML seems too many a revelation, a more extensive examination reveals that XML is much more complicated than it is generally made out to be, and that far from being a cure-all, it may end up being worse than the disease.

It is the intent of this paper not to be impartial, but rather to at least partially expose the many flaws of XML. To do that we will examine the nature of XML and its relationship to HTML and SGML; we will examine the nature of data itself, and how that relates to the measures taken to guarantee the validity of XML documents; and we will examine the uses XML is being put to, and whether XML is the appropriate tool for the job.

The Nature of XML

According to Ronald Schmelzer, XML is a way "to represent structured information in a text-based document" (Schmelzer, 2001, p. 45). The reason this is true is because it, like HTML, (Hyper-Text Markup Language,) is a subset of SGML, (Standardized General Markup Language.) A markup language is a way to handle text documents. By surrounding, (marking up,) the various elements with tags, the various pieces of a document can be easily identified. For example, the title of this paper could be marked up as follows: <title> The Case Against XML</title>. Thus the title of this paper would be clearly separated from the rest of the paper, and its semantic content would be clearly identified. We could go further still and identify the font to be used, the size, and whether the font was regular, bold, italicized, underlined, or any combination of the four choices. Each of these choices would encapsulate the text between the opening and closing tags.

HTML is the language used to create web pages. By encapsulating the various text elements with tags, the browser is able to determine the best way to display them. This works quite well if all we are sharing is text based documents, but it gets much more difficult when we are displaying pictures, playing video, or listening to sounds. HTML was extended so that it could manage all manner of binary data through the built in capabilities of the browser, or through

plug-ins that worked in conjunction with the browser. Thus when you click on a Microsoft ® Word document, a Word plug-in opens inside the browser, and the document displays inside the plug-in.

The problem with HTML is that the author lost control over the way the document was represented. While the author might choose to identify a text element as a title, the title tags might be processed and displayed differently by different web browsers. A corporation spends many thousands of dollars on creating and maintaining a certain image, and needed to be able to guarantee that image would be displayed exactly the same way, every time, everywhere. Likewise a document with extensive charts and graphs could not be guaranteed to display properly on everyone's machines.

This problem was solved through the used of a variety of scripting languages, through which the document content can be more precisely controlled, and through the conversion of documents into Adobe® Acrobat format, which in essence takes a picture of the pages. These pages can then be viewed exactly as intended by anyone using the free Adobe® Acrobat Reader.

Another method was to use meta data. Meta data is defined by Raggett, Hors & Jacobs (1999) " information about a document rather than document content" (Raggett, Hors & Jacobs, 1999). In other words, meta data contains semantic content about the representation

of the document content. Meta data is contained in the head of the HTML page, and can be used to give the author more precise control over the way the document is displayed. The assumption is made that the relationship between the document meta data and the document content will remain the same. Dr. Brooks describes this as "The False Document Assumption," whereby an assumption is made that "the semantic content of the <HEAD> element will maintain a time-invariant relationship with the semantic content of the <BODY> element" (Brooks, 2001, p. 7). But I have the ability to modify the document content separately from the display properties and values, thus destroying the supposed one-to-one relationship. Thus the use of meta data only works if the document is static, which the web is not (Brooks, 2001, p. 7)

To view an example of meta data for yourself, open any Word document, then save it into HTML format. Now on the menu bar select View/HTLM source. If you are using Word 97 or later, you will see at the top an extensive series of meta tags "declaring a property and a value for that property" (Raggett, et al.). These values can also be declared outside the document through the use of the "Link" element, as demonstrated in Figure 1.

```
<meta http-equiv=Content-Type content="text/html;
charset=windows-1252">
<meta name=ProgId content=Word.Document>
<meta name=Generator content="Microsoft Word 10">
<meta name=Originator content="Microsoft Word 10">
<link rel=File-List href="Case%20Study%201_files/filelist.xml">
```

Figure 1: Meta Data

If you were attentive, you surely noticed the Link element in Figure 1 referenced an XML file. This XML file contains specific instructions as to how various elements of the document were to be displayed. (In earlier Word implementations, these elements were all defined by meta tags contained within the document head itself.) Another technique to control the style of a document is through the use of cascading style sheets, or CSS, and various scripting languages. CSS is a powerful tool by which a style can be defined once, they used again and again for subsequent documents.

Before continuing on, you really need to turn to and glance through Appendix A, which is a complete example of the meta information for a 12 page, HTML formatted Word document. One thing you will no doubt discover is that Appendix A takes 7 single spaced pages to describe the content of the 12 page paper. You should also note that the some of the properties and their values are contained in a second document, linked to the first, making this an incomplete listing of the document properties. Also, although we haven't discussed them yet, the meta tags reference XML DTDs and

Schemas which are used to define the document properties and values. Thus the various methods used to control and maintain the format may often be more extensive than the document being described.

But with all these elements working together, we still do not have total control over the format of the Word document. Some browsers are not supported, and some computers may not contain the exact fonts required. Other difficulties may also present themselves. Therefore if we are to ensure our document is adequately represented on a distant machine, we must have a more powerful way to describe and validate the document contents. As quoted earlier, Schmelzer says this is the precise purpose of XML: it is designed "to represent structured information in a text-based document" (Schmelzer, 2001, p. 45).

But XML is more than just a markup language. It is eXtensible, meaning a person can use XML to define their own tags and thereby create their own languages, their own subsets of XML. These collections of tags can then be shared and plugged in to XML, thus allowing for the exact resonstruction of the document. These individual extensions to XML were originally called Document Type Definitions, (DTDs,) but are now giving way to XML Schemas.

In the version version 1.0 specification for XML, Bray, Paoli, Sperberg-McQueen, & Maler (2000) define a DTD as a set of "mark-up declarations that provide a grammar for a class of documents" (Bray, Paoli, Sperberg-McQueen, & Maler, 2000). The purpose of the DTD is to "define constraints on the logical structure and to support the use of predefined storage units" (Bray, et al.). This logical structure is not defined in the XML specification, as noted by Fabian Pascal, because the developers of XML did not have a data model in mind (Pascal, 2003). The eventual model was hierarchal, with a structure known as trees. According to Victor Vianu, the point of the XML data model is to ensure the set of data and its structure is one and the same (Vianu, 2001, p. 1).

The Nature of Data

To understand the problems with the XML data model, we must first define what data is. In their book "Database Systems," Rob and Coronel define data as "raw facts" (Rob & Coronel, 2002, p.6). But this is not enough. We must also have a data model, as the model provides the understanding of how raw data can be processed into information. Fabian Pascal writes that a data model must answer three simple questions: what things are, how they are related, and how to deal with them (Pascal, 2003). To be more precise, Mr. Pascal states that "a **data model** is a general theory of data that serves as a

"translator" for mapping enterprise-specific **business models** to enterprise-specific **logical models**. To that end, a **data model** provides four logical constructs---**data types, structure, integrity,** and **manipulation**" (Pascal, 2003; emphasis in the original).

Computers cannot understand business models; the end state of a data model is the production of the logical models computers do understand (Pascal, 2003). In the 1960s, Dr, Codd saw a need and developed the relational data model, (Date, 2003). The Relational Data Model is based on set math, predicate logic and dependency theory (Pascal, Feb, 2003). As you remember from grade school, sets are based on characteristics. The set of humans can be divided into two subsets of male and female, it can be divided up into subsets of race, it can be divided up into subsets of geography, national origin, age, or any other characteristic you choose. These various characteristics that create the subsets are called dimensions (Coronel, 2002, p. 642).

Predicates are declarative statements that are either true or false. "A person is male" is made up of the subject "A person" and the predicate "is male." We can create unions of the previous subsets through the use of the following predicates, or declarative statements: is male, is white, is South African, is of Dutch descent, is older than 18. (The dimensions are: gender, race, geography, national origin,

and age.) We have now created a union of sets based on particular data characteristics.

Notice what the use of the relational data model does: it breaks up entities into their most basic characteristics, and then allows them to be recombined in an ad hoc basis to create useful information out of what was raw data. The raw data is logically represented in the computer in a form it can recognize, but through the application of relational algebra---which is derived from set math, predicate logic & dependency theory---the information can be extracted from its logical form and translated into whatever business model is appropriate (Date, 2001). Thus the physical memory locations and the logical structure are separated from the business model. The level of abstraction is such that a user can specify what is to be done, but not how it is to be done.

XML turns all that on its ear. Whereas the model for relations is based on math, logic and theory, the XML model is based on structures, particularly hierarchal tree structures. As Victor Vianu says, "A schema…constrains the paths…that can be found in the data graph (Vianu, 2001, p. 3). Note the problem: the business model and the logical structure are bound together. This makes ad hoc queries difficult, as the tree is designed to be traversed in a particular order. An example would be a hierarchal structure for an aircraft supply

warehouse. An aircraft is made up of systems, which are made up of sub systems, which are made up of individual assemblies, which are themselves made up of individual subassemblies, which are made up of individual parts. This tree structure works well if I want to ask what parts comprise a particular assembly; I simply traverse all the nodes of the tree until I reach the individual leaves, (parts,) then list the parts.

But what happens if I want to find out where the 3/16" x 1 1/2" left hand thread bolt is used? I have to traverse all the nodes down to the leaves to find that information; this is a time consuming process. But the reliance on this hierarchal construct has implications for the real world as well. Since each assembly is made up of certain parts, the warehouse must maintain a stock of those parts. Since the hierarchal structure does not lend itself to listing the assemblies a certain part is used on, it is simpler to maintain separate stocks of items for each assembly. Thus a separate stock of 3/16" x 1 1/2" left hand thread bolts is maintained for each assembly it is used on. This is clearly wasteful in space, and money is tied up on duplicate inventory. Also, if we are out of the bolt used for a particular application, we have no way of knowing that the same bolt is stored other places in the warehouse, creating a supply emergency where none need have existed.

This glaring error quickly became apparent, and so the XML data schema was extended to "decouple the element names from their types and thus allow context-dependent definitions of their structure" (Vianu, 2001, p. 4). Although the Mr. Vianu uses a lot of big words and high sounding concepts to describe this, it is clear from the New/Used car ad example he gives that these extended schemas simply involve the creation of multiple trees, each of which is chosen on the fly based on the context. As we have demonstrated in Appendix 4, the markup of a simple document can be nearly as large as the document itself. Now we are faced with extended schemas containing multiple trees. This is not only a storage and transmission burden, but the actual processing of the trees must be done by the client and not on some dedicated machine. This is exceedingly inefficient.

I mentioned earlier that XML required a hierarchal, tree based structure. Hierarchies are created through a parent/child relationship, in which a parent can have many children, but a child can have only one parent (Coronel, 2002, p. 26). In database terms, this is a one-to-many relationship. But life does not consist of nice, neat one-to-many relationships. A class is made up of many students. Many students take many classes. Hierarchal structures cannot readily represent this relationship (Coronel, 2002, p. 28).

XML and its Uses

XML is based on SGML. Thus XML is really a language to describe documents. The World Wide Web Consortium, (W3C,) created the Document Object Model to "process XML documents for representation as a data bound tree" (Schmelzer, p. 6). The individual data elements need not be physically located together for the document to be reconstructed; in fact, the pieces need not be in the same location. Dr. Brooks quotes Kurt Cagle as looking forward to the day when "a single XML 'document' could conceivably span hundreds or thousands of servers" (Brooks, 2001). This distributed data is perhaps the genesis of the much awaited XML database (Pascal, 2003, Setting Matters Straight). Dr. Brooks points out that meaning is not necessarily tied to form, but "resides in the semantic structure of the information" (Brooks, 2001). Schmelzer squelches this by pointing out that "XML tools that reference external resources introduce some processing unpredictability. The main problem is that it is difficult to predict when and even if those resources will be available. [These dependencies] add an unknown amount of time to the work" (Schmelzer, 2001, p. 5).

An example of the problem of distributed dependencies arose while I was researching to find the source document of the Kurt Cagle essay quoted in Dr. Brooks article. While I was able to find the home

page for Kurt Cagle, (http://www.vbxml.com/members/profile.asp?id=i3392), I was unable to view the source document due to some sort of processing error. Should such a thing to happen during the recreation of an XML document, a portion of the document would be missing. The opposite could also be true, of course--- storing a document over multiple servers may guarantee that in the event of an unfortunate event, at least some of the document can be recovered.

XML is primarily being mentioned today as a technology for data exchange. But XML is a presentation technology (Pascal, 2003, The Data Exchange Tail). XML is useful for presenting data, but not for infering its meaning. Data exchange, however is not concerned with the way the data is presented, but with its meaning. "Data exchange requires agreement on (a) what data is to be exchanged, and (b) its physical format, which are orthagonal (independent) considerations" (Pascal, 2003, The Data Exchange Tail). Two or more parties agree upon a common meaning for the data. All that is left is to determine the formatting of the data. If XML is chosen, the data is encapsulated with tags describing its characteristics. This is unneccesarily redundant, as the meaning had already been agreed upon. Dr. Pascal remarks that "for data exchange purposes the tags are no more than delimiters, just like commas, or spaces, or any other

such that tells the receiving system where data values physically start and end" (Pascal, 2003, The Data Exchange Tail).

Since XML tags are delimiters, the question becomes whether their use adds value. The case can be made that an XML delimited document is readable and self describing, something that cannot be said with an ASCII delimited file. But XML is an extremely verbose format for data exchange. The tags overwhelm the data, as the example in Figure 2 shows (Pascal, 2003, The Data Exchange Tail). While bandwidth is not the concern it once was, the effects such a verbose format has on the network infrastructure must be considered.

```
-<File_Info>
<Filename_Versioned>xlplot.zip</Filename_Versioned>
<Filename_Previous>xlplot.zip</Filename_Previous>
<Filename_Generic>xlplot.zip</Filename_Generic>
<Filename_Long />
<File_Size_Bytes>2697970</File_Size_Bytes>
<File_Size_K>2570</File_Size_K>
<File_Size_MB>2.57</File_Size_MB>
</File_Info>
-<Expire_Info>
<Has_Expire_Info>N</Has_Expire_Info>
<Expire_Count />
<Expire_Based_On>Days</Expire_Based_On>
<Expire_Other_Info />
<Expire_Month />
<Expire_Day />
<Expire_Year />
</Expire_Info>
```

Figure 2: Tags Overwhelm Data (Pascal, 2003, The Data Exchange Tail).

Another advantage of using XML delimited data elements is that they need not be in any particular order. The XML document is self-desribing, will be parsed at the receiving end, and will have its

structure recreated according to the needs of the receiver. ASCII delimited files, on the other hand, contain data elements that must be in a previously agreed upon order. If one comma is misplaced, or if one data element is missing, the meaning of the file from that point on is lost.

As a format for data exchange, XML does not provide "security, authentication, privacy, reliability, and traceability" (Schmelzer, 2001, p. 5). All these concerns must be addressed using other standards, media, and protocols. Proponents of Electronic Data Interchange, (EDI,) point out that this mature format supports all these concerns, and that adding the mechanisms onto an XML document to provide the same benefits as EDI ends up costing as much as EDI (Schmelzer, 2001, p.5). So where is the benefit?

Conclusion

XML is a document oriented markup language designed to control the presentation of that document. It is not being pressed into uses for which it was never designed and is ill-equipped. XML does not have a data model, but does have a data structure: the tree. Trees, or hierarchies are ill-equipped to deal with the complexities of the world. Many ordinary relationships cannot be adequately expressed in tree form.

Since no one XML hierarchal structure is useful in all situations, XML has been extended such that multiple trees are created, with the particular tree to be used chosen based on the context. Thus the semantic content of a document often equals or exceeds the actual size of the document. This means that documents are larger than necessary.

XML has severe limitations as a data exchange formate. While in its native form XML may be less expensive than alternatives like EDI, once you factor in the cost of the technologies for security, privacy, authentication, etc., the cost becomes much the same. XML encapsulates data with delimiting tags that describe its meaning. This is redundant, as the meaning has already been agreed upon. What XML is, then, is one possible format, and an extremely verbose one at that. While the readability and self-describing nature of XML do make it somewhat attractive, the data is overwhelmed by the tags used to describe it. Thus XML is not a good shoice for data exchange.

I question the need for XML as a presentation technology. We have many presentation formats in use today, and some of them like Adobe Acrobate are becoming universal standards. What is needed is not another presentation format, but a way to ensure the document is transmitted and viewed exactly as intended. Adobe Acrobat does that

with ease, and without the fuss and bother of special editors and parsers.

XML is a new technology, and may yet prove useful. It would appear, however, that it has several flaws, flaws which have been glossed over, and flaws that are getting worse instead of better. If I was a betting man, I would guess the flaws of XML will become apparent as it becomes more widespread, and that it will eventually shrivel away into a niche technology.

References

Bray, T., Paoli, J., Sperberg-McQueen, C. M., & Maler, E. (2000). Extensible Markup Language (XML) 1.0 (Second Edition). Retrieved July 7, 2003, from http://www.w3.org/TR/Rec-xml

Brooks, T. A. (n.d.). Advantages of XML. Retrieved June 14, 2003 from the University of Washington Web site: http://faculty. washington.edu/tabrooks/XML/ XMLTheory.htm

Brooks, T. A. (2001). Where is meaning when form is gone? Knowledge representation on the Web. Information Research, 6, 2. Retrieved June 14, 2003 from http://informationnr.net/ir/6-2/paper93.html

Budelman, R. (n.d.). India XML: The Future. Retrieved July 7, 2003 from India Web Developers Web site: http://www. indiawebdevelopers.com/technology/ XML/xml.asp

Date, C. J. (2001). Constraints and Predicates: A Brief Tutorial Part II. Retrieved July 8, 2003 from http://www.pgro.uk7.net/cjd7a2.htm

Date, C. J. (2003). Dr. Edgar F. Codd (1923-2003): An Appreciation. Retrieved July 8, 2003. Available at http://www.dbdebunk.com/

Pascal, F. (2003). The Data Exchange Tail – Part 2. DBAzine.com. Retrieved July 8, 2003 from http://www.dbazine.com/pascal9.htm

Pascal, F. (2003). Tags Do Not a Language Make – Part 1. DBAzine.com. Retrieved June 14, 2003, from http://www.dbazine.com/pascal8.html

Pascal, F. (Feb 2003). On Database Fundamentals. Retrieved July 8, 2003 from http://www.pgro.uk7.net/gw1.htm

Pascal, F. (Apr 2003). Setting Matters Straight – April 2003. Retrieved June 14, 2003 from http://www.tdan.com/sms_issue24.htm

Raggett, D., Hors, A. L., & Jacobs, I. (1999). HTML 4.01 Recommendation. Retrieved July 7, 2003, from http://www.w3.org/TR/REC-html40/cover.html#minitoc

Rob, P. & Coronel, C. (2002). Database Systems: Design, Implementation, and Management (5th ed.). Boston, MA: Thompson Learning

Schmelzer, R. (2001). The Pros and Cons of XML. PDF file retrieved June 26, 2003 and available from http://www.zapthink.com/reports/proscons-view.html

Thomas, J. P. (n.d.). Pros & Cons of XML. Retrieved June 26, 2003 from India Web Developers Web site: http://www.

indiawebdevelopers.com/technology/

XML/xml_advan.asp

Thomas, J. P. (n.d.). XML (eXtensible Markup Language) ...technology of choice. Retrieved July 7, 2003 from India Web Developers Web site: http://www.indiawebdevelopers.

com/technology/XML/xml_lang.asp

Vianu, V. (2001). A Web Odyssey: from Codd to XML. Retrieved June 14, 2003.

Available from http://80-delivery.acm.org.ezproxy.umuc.edu/10.1145/380000

/375554/p1-vianu.pdf?key1=375554&key2=5541065501&coll=portal&dl=

ACM&CFID=10919671&CFTOKEN=16538804

ZapThink, LLC. (2001). XML Expertise Put to Work for You: ZapThink Analysis & Research. In The Pros & Cons of XML (pp. 47-52). PDF file retrieved June 26, 2003 and available from http://www.

zapthink.com/reports/proscons-view.html

Appendix A

Meta Tags in a Word Document Saved in HTML format

```
<html xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:w="urn:schemas-microsoft-com:office:word"
xmlns:st1="urn:schemas-microsoft-com:office:smarttags"
xmlns="http://www.w3.org/TR/REC-html40">

<head>
<meta http-equiv=Content-Type content="text/html;
charset=windows-1252">
<meta name=ProgId content=Word.Document>
<meta name=Generator content="Microsoft Word 10">
<meta name=Originator content="Microsoft Word 10">
<link rel=File-List href="Case%20Study%201_files/filelist.xml">
<title>Case Study 1: National Logistics Management</title>
<o:SmartTagType namespaceuri="urn:schemas-microsoft-com:office:
smarttags"
 name="PlaceType"/>
<o:SmartTagType namespaceuri="urn:schemas-microsoft-com:office:
smarttags"
 name="place"/>
<!--[if gte mso 9]><xml>
 <o:DocumentProperties>
  <o:Author>Preferred Customer</o:Author>
  <o:LastAuthor>Preferred Customer</o:LastAuthor>
  <o:Revision>2</o:Revision>
  <o:TotalTime>1446</o:TotalTime>
  <o:LastPrinted>2003-07-06T13:48:00Z</o:LastPrinted>
  <o:Created>2003-07-06T21:09:00Z</o:Created>
  <o:LastSaved>2003-07-06T21:09:00Z</o:LastSaved>
  <o:Pages>1</o:Pages>
  <o:Words>2421</o:Words>
  <o:Characters>13804</o:Characters>
  <o:Company>Dell Computer Corporation</o:Company>
  <o:Lines>115</o:Lines>
  <o:Paragraphs>32</o:Paragraphs>
  <o:CharactersWithSpaces>16193</o:CharactersWithSpaces>
  <o:Version>10.4219</o:Version>
 </o:DocumentProperties>
</xml><![endif]--><!--[if gte mso 9]><xml>
 <w:WordDocument>
  <w:Zoom>0</w:Zoom>
```

```
  <w:BrowserLevel>MicrosoftInternetExplorer4</w:BrowserLevel>
 </w:WordDocument>
</xml><![endif]--><!--[if !mso]><object
 classid="clsid:38481807-CA0E-42D2-BF39-B33AF135CC4D"
id=ieooui></object>
<style>
st1\:*{behavior:url(#ieooui) }
</style>
<![endif]--><![if !supportAnnotations]>
<style id="dynCom" type="text/css"><!-- --></style>
<script language="JavaScript"><!--
function msoCommentShow(anchor_id, com_id)
{
        if(msoBrowserCheck())
                {
                c = document.all(com_id);
                a = document.all(anchor_id);
                if (null != c && null == c.length && null != a && null ==
a.length)
                        {
                        var cw = c.offsetWidth;
                        var ch = c.offsetHeight;
                        var aw = a.offsetWidth;
                        var ah = a.offsetHeight;
                        var x  = a.offsetLeft;
                        var y  = a.offsetTop;
                        var el = a;
                        while (el.tagName != "BODY")
                                {
                                el = el.offsetParent;
                                x = x + el.offsetLeft;
                                y = y + el.offsetTop;
                                }
                        var bw = document.body.clientWidth;
                        var bh = document.body.clientHeight;
                        var bsl = document.body.scrollLeft;
                        var bst = document.body.scrollTop;
                        if (x + cw + ah / 2 > bw + bsl && x + aw - ah / 2 -
cw >= bsl )
                                { c.style.left = x + aw - ah / 2 - cw; }
                        else
                                { c.style.left = x + ah / 2; }
                        if (y + ch + ah / 2 > bh + bst && y + ah / 2 - ch >=
bst )
                                { c.style.top = y + ah / 2 - ch; }
```

```
                    else
                        { c.style.top = y + ah / 2; }
                    c.style.visibility = "visible";
}       }       }
function msoCommentHide(com_id)
{
        if(msoBrowserCheck())
                {
                c = document.all(com_id);
                if (null != c && null == c.length)
                {
                c.style.visibility = "hidden";
                c.style.left = -1000;
                c.style.top = -1000;
                } }
}
function msoBrowserCheck()
{
        ms = navigator.appVersion.indexOf("MSIE");
        vers = navigator.appVersion.substring(ms + 5, ms + 6);
        ie4 = (ms > 0) && (parseInt(vers) >= 4);
        return ie4;
}
if (msoBrowserCheck())
{
        document.styleSheets.dynCom.addRule(".msocomanchor",
"background: infobackground");
        document.styleSheets.dynCom.addRule(".msocomoff","display:
none");
        document.styleSheets.dynCom.addRule(".msocomtxt","visibility:
hidden");
        document.styleSheets.dynCom.addRule(".msocomtxt","position:
absolute");
        document.styleSheets.dynCom.addRule(".msocomtxt","top: -
1000");
        document.styleSheets.dynCom.addRule(".msocomtxt","left: -
1000");
        document.styleSheets.dynCom.addRule(".msocomtxt","width:
33%");
        document.styleSheets.dynCom.addRule(".msocomtxt",
"background: infobackground");
        document.styleSheets.dynCom.addRule(".msocomtxt","color:
infotext");
        document.styleSheets.dynCom.addRule(".msocomtxt","border-
top: 1pt solid threedlightshadow");
```

```
        document.styleSheets.dynCom.addRule(".msocomtxt","border-
right: 2pt solid threedshadow");
        document.styleSheets.dynCom.addRule(".msocomtxt","border-
bottom: 2pt solid threedshadow");
        document.styleSheets.dynCom.addRule(".msocomtxt","border-
left: 1pt solid threedlightshadow");
        document.styleSheets.dynCom.addRule(".msocomtxt","padding:
3pt 3pt 3pt 3pt");
        document.styleSheets.dynCom.addRule(".msocomtxt","z-index:
100");
}
// --></script>
<![endif]>
<style>
<!--
 /* Font Definitions */
 @font-face
        {font-family:Courier;
        panose-1:2 7 4 9 2 2 5 2 4 4;
        mso-font-alt:"Courier New";
        mso-font-charset:0;
        mso-generic-font-family:modern;
        mso-font-format:other;
        mso-font-pitch:fixed;
        mso-font-signature:3 0 0 0 1 0;}
@font-face
        {font-family:Tahoma;
        panose-1:2 11 6 4 3 5 4 4 2 4;
        mso-font-charset:0;
        mso-generic-font-family:swiss;
        mso-font-pitch:variable;
        mso-font-signature:1627421319 -2147483648 8 0 66047 0;}
 /* Style Definitions */
 p.MsoNormal, li.MsoNormal, div.MsoNormal
        {mso-style-parent:"";
        margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:12.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        font-size:12.0pt;
        font-family:"Times New Roman";
        mso-fareast-font-family:"Times New Roman";}
h1
        {mso-style-next:Normal;
```

```
        margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:3.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        page-break-after:avoid;
        mso-outline-level:1;
        font-size:16.0pt;
        font-family:Arial;
        mso-font-kerning:16.0pt;}
h2
        {mso-style-next:Normal;
        margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:3.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        page-break-after:avoid;
        mso-outline-level:2;
        font-size:14.0pt;
        font-family:Arial;
        font-style:italic;}
h3
        {mso-style-next:Normal;
        margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:3.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        page-break-after:avoid;
        mso-outline-level:3;
        font-size:13.0pt;
        font-family:Arial;}
h4
        {mso-style-next:Normal;
        margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:3.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        page-break-after:avoid;
        mso-outline-level:4;
        font-size:14.0pt;
        font-family:"Times New Roman";}
h5
```

```
        {mso-style-next:Normal;
        margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:3.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        mso-outline-level:5;
        font-size:13.0pt;
        font-family:"Times New Roman";
        font-style:italic;}
p.MsoCommentText, li.MsoCommentText, div.MsoCommentText
        {mso-style-noshow:yes;
        margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:12.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        font-size:10.0pt;
        font-family:"Times New Roman";
        mso-fareast-font-family:"Times New Roman";}
p.MsoHeader, li.MsoHeader, div.MsoHeader
        {margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:12.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        tab-stops:center 3.0in right 6.0in;
        font-size:12.0pt;
        font-family:"Times New Roman";
        mso-fareast-font-family:"Times New Roman";}
p.MsoFooter, li.MsoFooter, div.MsoFooter
        {margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:12.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        tab-stops:center 3.0in right 6.0in;
        font-size:12.0pt;
        font-family:"Times New Roman";
        mso-fareast-font-family:"Times New Roman";}
span.MsoCommentReference
        {mso-style-noshow:yes;
        mso-ansi-font-size:8.0pt;
        mso-bidi-font-size:8.0pt;}
```

```
p.MsoBodyTextIndent2, li.MsoBodyTextIndent2, div.
MsoBodyTextIndent2
        {margin-top:0in;
        margin-right:0in;
        margin-bottom:0in;
        margin-left:58.5pt;
        margin-bottom:.0001pt;
        text-indent:-27.0pt;
        mso-pagination:none;
        font-size:10.0pt;
        font-family:Courier;
        mso-fareast-font-family:"Times New Roman";
        mso-bidi-font-family:Courier;}
p.MsoAutoSig, li.MsoAutoSig, div.MsoAutoSig
        {margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:0in;
        margin-left:0in;
        margin-bottom:.0001pt;
        mso-pagination:widow-orphan;
        font-size:12.0pt;
        font-family:"Times New Roman";
        mso-fareast-font-family:"Times New Roman";}
p.MsoCommentSubject, li.MsoCommentSubject, div.
MsoCommentSubject
        {mso-style-noshow:yes;
        mso-style-parent:"Comment Text";
        mso-style-next:"Comment Text";
        margin-top:12.0pt;
        margin-right:0in;
        margin-bottom:12.0pt;
        margin-left:0in;
        mso-pagination:widow-orphan;
        font-size:10.0pt;
        font-family:"Times New Roman";
        mso-fareast-font-family:"Times New Roman";
        font-weight:bold;}
 /* Page Definitions */
 @page
        {mso-mirror-margins:yes;}
@page Section1
        {size:8.5in 11.0in;
        margin:1.0in 1.0in 1.0in 1.5in;
        mso-header-margin:.5in;
        mso-footer-margin:.5in;
```

```
        mso-header:url("Case%20Study%201_files/header.htm") h1;
        mso-paper-source:0;}
div.Section1
        {page:Section1;}
-->
</style>
<!--[if gte mso 10]>
<style>
 /* Style Definitions */
 table.MsoNormalTable
        {mso-style-name:"Table Normal";
        mso-tstyle-rowband-size:0;
        mso-tstyle-colband-size:0;
        mso-style-noshow:yes;
        mso-style-parent:"";
        mso-padding-alt:0in 5.4pt 0in 5.4pt;
        mso-para-margin:0in;
        mso-para-margin-bottom:.0001pt;
        mso-pagination:widow-orphan;
        font-size:10.0pt;
        font-family:"Times New Roman";}
table.MsoTableGrid
        {mso-style-name:"Table Grid";
        mso-tstyle-rowband-size:0;
        mso-tstyle-colband-size:0;
        border:solid windowtext 1.0pt;
        mso-border-alt:solid windowtext .5pt;
        mso-padding-alt:0in 5.4pt 0in 5.4pt;
        mso-border-insideh:.5pt solid windowtext;
        mso-border-insidev:.5pt solid windowtext;
        mso-para-margin-top:12.0pt;
        mso-para-margin-right:0in;
        mso-para-margin-bottom:12.0pt;
        mso-para-margin-left:0in;
        mso-pagination:widow-orphan;
        font-size:10.0pt;
        font-family:"Times New Roman";}
</style>
<![endif]--><!--[if gte mso 9]><xml>
 <o:shapedefaults v:ext="edit" spidmax="3074"/>
</xml><![endif]--><!--[if gte mso 9]><xml>
 <o:shapelayout v:ext="edit">
  <o:idmap v:ext="edit" data="1"/>
 </o:shapelayout></xml><![endif]-->
```

```
</head>
```