RelaxNG with XML Data Structures

Dave Kennedy
Christchurch Polytechnic Institute of Technology
Christchurch, NZ
kennedyd@cpit.ac.nz

ABSTRACT

XML mark-up is used in a wide range of applications and in particular for data transfer via the Internet. XML textbooks and web-based tutorials typically introduce XML mark-up by the use of examples and the rules for well-formed XML, e.g. XML Tutorial. According to this popular presentation at W3schools.com the development of a DTD or schema is done after a document instance has been marked up. An XML document is a data structure. This paper proposes a methodology for XML mark-up that begins with data analysis by using a type of structure diagram (an elm tree diagram) then describes the data structure and content using Relax NC (an easier to use alternative to DTD or XML Schema) and finally does the mark-up of document instances. The instruction effectiveness and the efficiency of the different approaches are compared for two occurrences of a 3rd year degree paper.

Keywords

elm tree diagram, Relax NG, XML Schema, XML data structures.

1. INTRODUCTION

Gaven (2002) is desperately seeking a set of simplified best practices and modelling ideas for XML data structures. Mable (2002) is looking for a DTD and Schema editor akin to an ER tool which would allow a developer to build XML schemas. I teach our students about XML. The students mark up a course descriptor as an XML document, create a DTD and schema, and write xsl code that reproduces the original document; I am often surprised at the range of different solutions, some are definitely better than others. The cries from the students are “Where do I start? What am I trying to do?” They too would like a set of best practices and modelling ideas for XML data structures.

The problems:

- ♦ How best to teach XML markup?
- ♦ What are the recognised best practices?
- ♦ Is there a suitable methodology for creating XML data structures?

RelaxNC (Cowan & Murata, 2001) appears to be a large part of the answer to all of these problems. RelaxNC is a human-friendly language for describing an XML schema (Fitzgerald, 2002). It is possible to “think” in RelaxNC and to build the schema for a document **before** doing the markup. While working through some examples I found myself sketching structure diagrams before writing the RelaxNC. Such a structure diagram approach has been developed for modelling SGML structure and can equally be applied to the representation of XML structures. Elm (enables lucid models) tree diagrams are a formal structure diagram approach to modelling document structure and content (Maler & el Andaloussi, 1996).

A summer school occurrence of our client-side programming paper was an opportunity to try a different approach to the teaching of XML. This time the assignment was to create the elm tree diagram for a course outline, then the RelaxNC schema **before** marking up a particular instance. The xsl section was unchanged i.e. write xsl to recreate the original document. The results were most encouraging. The students learnt the techniques more quickly, produced better xml structures and consequently wrote more efficient xsl code.

2. BEST PRACTICE AND MODELLING IDEAS

Data modelling is an art as much as a science and building XML data structures is no exception. It is possible to produce a wide range of different XML mark-up for the one document. The question is not only what is good XML mark-up but also how do you do this mark-up. The following 9 rules describe XML mark-up best practices and a methodology for the analysis and design of XML data structures.

2.1 Do the data analysis first

Many introductory texts introduce XML by working through a document and marking up the content using the XML rules for well-formed XML and applying appropriate semantic tags. This approach presents XML mark up as another form of HTML i.e. simply adding tags to a document.

XML is a mark-up for content. An XML document is a data structure and in this sense all XML documents are data-centric even if some of the data is a paragraph rather than a discrete number or word. Accordingly it makes sense to do some data analysis before rushing in and marking up the content. As St Laurent (2002) notes "The successful application of XML requires data modelling experience". However XML mark-up is more than just data modelling. It is a mark-up for both content and structure (St Laurent, 2002). To do this you have to separate the structure from the content (Patrizio, 2002; St Laurent, 1998).

How best to do this analysis? (What are the rules? How do you model an XML structure? What tools are available?)

If you mark-up a single sample document and then create the corresponding DTD the incidental structure of the XML document largely determines the structure of the related schema or DTD and this has a major influence on the subsequent processing code. Given an XML instance, tools such as XMLSpy can create the corresponding DTD or XML Schema. i.e. the mark-up determines the schema. From my experience of marking up documents and from teaching students XML technologies I would contend that it is better to create the schema first and then do the mark-up of documents.

Although XML is touted as easily human- readable, and indeed this is one of the w3c goals for XML, a more important goal is to make it easily machine-readable i.e. easy to write programs that will read and manipulate it e.g. xsl code (Baker, 2001, w3C Recommendation, 2000). Baker (2001) makes the point that "XML is for machines not people". In contrast to the human readable goal the W3 Communications Team (2001) state that "XML is text, but it is not meant to be read".

XML is about marking up the data. This implies that data analysis would be a useful thing to do first. Resist the temptation to simply start at the top of the document and mark up the data as you come to it (Muench, 2002). Instead

a) identify the components

b) classify the components into logical groups

(Maler & el Andaloussi, 1996, p. 30; XML Patterns).

However, as noted above, the data analysis for XML is more than just about identifying the components. It is also about defining the structures – how many times can this element occur?, is it optional?, how are elements nested?, is order important?

St Laurent (2002) makes the point that this is not your usual data analysis – "it is about using mark-up to identify rather than constrain – about thinking how *best* to represent the information using XML – it is more about *painting* on the structure rather than breaking the data into strictly pre-defined pieces".

Figure 1 shows the proposed methodology.

2.2 Code only the data

Leave any presentation elements for the style sheets. XML is all about separating the data from the presentation (St Laurent, 1998; Morrison et al, 2000, p. 13; Hoenisch, 2001).

"In well designed XML applications the mark-up says nothing about how the document should be displayed." (Harold & Means, 2001, p. 4).

For example:

```
yield =”60 bars”
```

and not

```
yield=”Recipe yields 60 bars (1 per
 serving)”
```

and not

```
<title><b>Anise-Almond Biscotti
   </b></title>
```

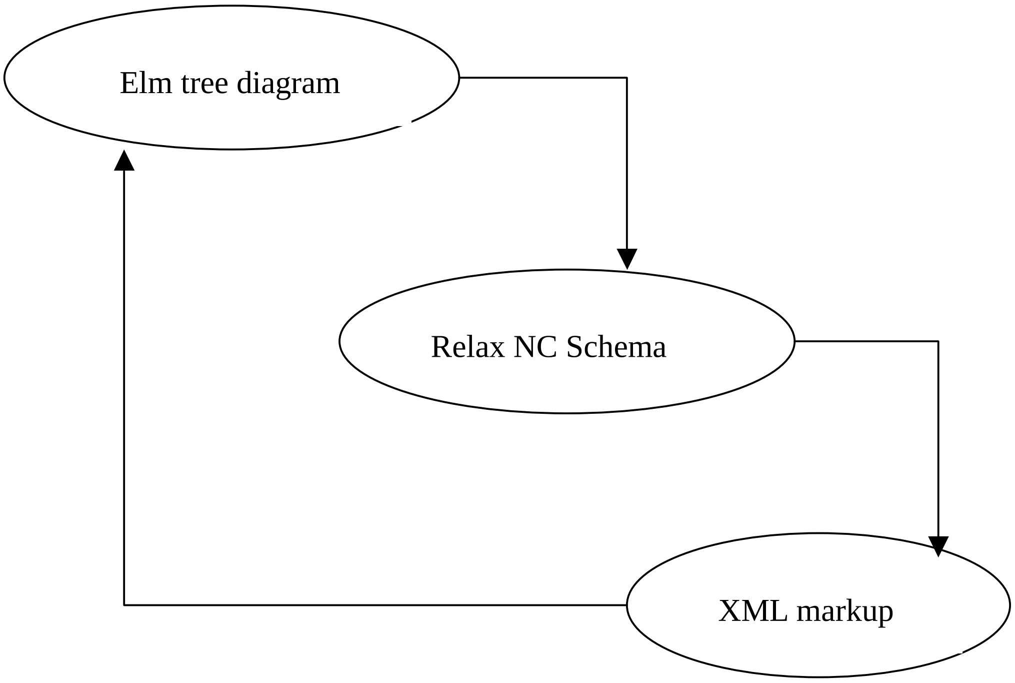


Figure 1. An XML mark-up methodology.

The XML should not be identifying bold or italics or the font size but rather why this data may sometimes be output in bold or in a large font.

2.3 Use collection elements

If an element can appear multiple times at the same level then create a collection element for them (XML Patterns; Maler & el Andaloussi, 1996, p. 102; Daconta, 2001).

```
e.g.  <ingredients>
   <ingredient>margarine</ingredient>
   <ingredient>flour</ingredient>
   ...
   </ingredients>
```

This makes the XML more easily human-readable but more importantly more machine-readable as each collection can be processed with a loop instruction (e.g. xsl:for-each).

2.4 Use a group or block element to control the sequence of elements and occurrences of elements or groups of elements.

(Ray, 2001, p. 59; Mahler & el Andaloussi, 1996, p. 102; Jelliffe, 1998, p. 1-59; Daconta, 2001; XML Patterns)

```
e.g. <recipe>
   <ingredients>
     ...
   </ingredients>
   <steps>
    ...
   </steps>
       </recipe>
```

XML mark-up is a tree structure but you don't want a tall tree or a wide hedge, it's best to aim for a wide, bushy shrub. (Ray, 2001, p. 171)

2.5 The best way to represent multi-valued data is via sub-elements.

The starting point for data analysis is identifying the components and their attributes. Multi-valued attributes are represented by a collection element (Daconta, 2001; XML Patterns).

```
e.g.      <book>
          <title>XML in a Nutshell</title>
          <authors>
                <author>E R Harold</
 author>
                <author>W S Means</
 author>
          </authors>
          </book>
```

2.6 Avoid "mixed" content

Mixed content means an element that contains both elements and text.

```
e.g.  <name>Dave <surname>Kennedy</
 surname></name>
```

The problem with mixed content is that the corresponding schema is too general to be any use in validating a document. Good schemas define the structure precisely and can be used to enforce it (Muench, 2002; Morrison *et al.*, 2000, p. 48).

It is better to use

```
<name>
    <firstName>Dave</firstName>
    <surname>Kennedy</surname>
</name>
```

2.7 Use meaningful names and a standard case format

Meaningful names make the document more human-readable especially if it is displayed by a program such as Internet Explorer. It also makes good programming sense, provides internal documentation, and makes de-bugging easier. XML tags are case-sensitive, it makes life much easier if you use a standard case format e.g. camelCase (Hoenisch, 2001) or lowercase only (Ray, 2001, p. 60) or dot notation (Carlson, 2001, pg.108).

2.8 Use attributes and elements as appropriate

Deciding whether to use XML attributes or elements is not as straightforward as it appears. Because elements are more familiar to people who know html and because you have to learn about elements before you can use an attribute, but not vice versa, XML structures tend to be element heavy (Daconta, 2001). But there are people who favour an attribute-heavy approach (Muench, 2002). An attribute-heavy data structure is not so human-friendly but it is more machine-readable (Muench, 2002).

This opens the debate on elements verses attributes. There appears to be much conflicting advice on this (St Laurent, 2002; Muench, 2002; XML Attributes; Harold & Means, 2001, p. 16). It is possible to write XML which is all elements and XML for the same document which is predominately attributes.

Ray (2001, p. 61), St Laurent (2002), and also Daconta (2001) advise that it is best to use elements for the data and attributes to hold the meta-data for an element. Jelliffe (1998, p. 1-96) uses a noun and adjective analogy. If it is like a noun use an element, if it is like an adjective use an attribute. Brandin (2003) argues that attributes should only be used to provide information about data elements in their scope.

```
   E.g. <ingredient meat=”N”
 aisle=”baking”>flour</ingredient>
```

The attribute-heavy approach appears to come from a database, data integrity, and data transfer viewpoint. It is also a consequence of the limited data typing available when using a DTD (Morrison *et al.*, 2000, pp. 47, 48).

It seems that the approach you take depends on the application (Harold & Means, 2001, p. 16). For data transfer it may well be that the more rigidly the structure is defined the better and an attribute-heavy approach with strict data typing is appropriate (Muench, 2002) but if you need the flexibility and extensibility that XML can provide, then an element-heavy structure with minimal data typing is more appropriate (Harold & Means, 2001, p. 16; St Laurent, 2002).

2.9 Recognise that defining the schema is an iterative process.

An XML structure is a data structure. Database analysis is an iterative process – often it is not until data is inserted into the tables that design flaws are exposed – and so the design is changed. Sometimes it is not until an application is written that design limitations are realised – and again the design is changed. So it is with XML (Jelliffe, 1998, p. 1-59). As it is with data base design – the more experience you have the more you are aware of downstream implications of your design; on data redundancy, on ease of programming, on performance – so it is with XML design. Baker (2001) sums it up nicely, “Think of the poor sap who has to do the programming”. This is a different consideration from the St Laurent (2002) maxim “Optimising mark-up for processing is always premature”. St Laurent is more concerned with constructing an XML structure that restricts possible variations only because it makes life easier for the programmers.

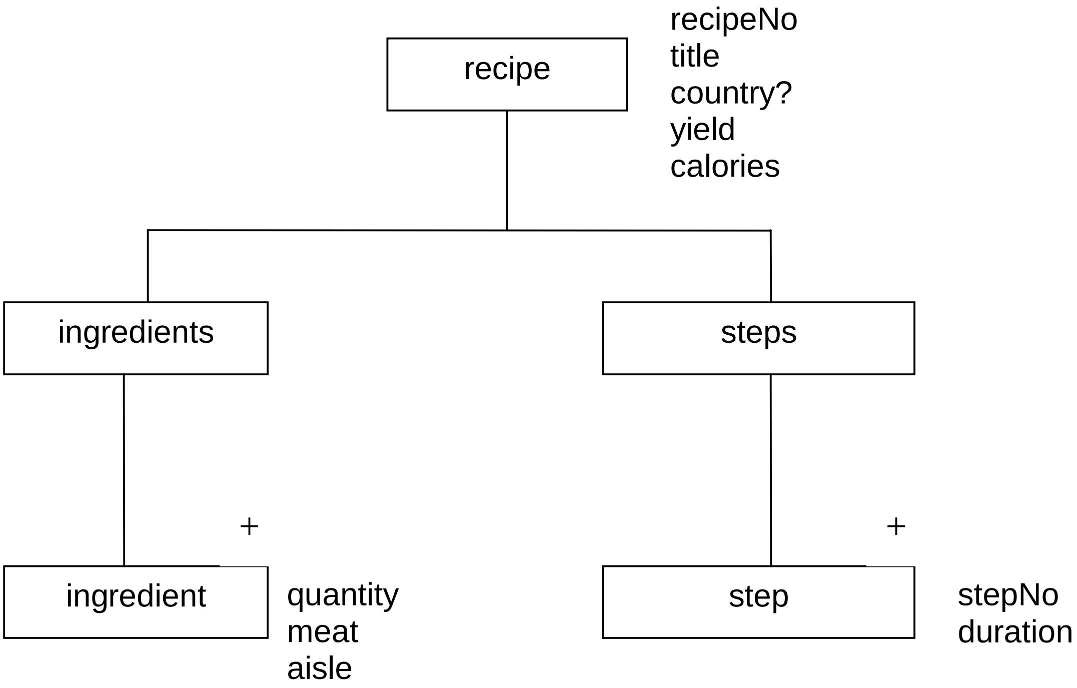


Figure 2. An elm tree diagram for the recipe example

3. ELM TREE DIAGRAM

With the above rules in mind a useful starting point for data analysis is an elm tree diagram (Maler & el Andaloussi, 1996).

Example: Assume we have a number of recipes similar in format to the one below.

```
Recipe Number: rec143
Anise-Almond Biscotti
Country of origin: Italy
Recipe yields 60 bars (1 per
      serving)
Calories per serving: 41

List of Ingredients
♦ 4 tablespoons margarine (meat: no,
aisle: dairy)
♦ 3/4 cup sugar (meat: no, aisle:
baking)
♦ 4 eggs (meat: no, aisle: dairy)
♦ 2 1/2 cups all-purpose flour (meat:
no, aisle: baking)
♦ 2 teaspoons crushed anise seeds
(meat: no, aisle: spices)
♦ 1 1/2 teaspoons baking powder (meat:
no, aisle: baking)
♦ 1/4 teaspoon salt (meat: no,
aisle: spices)
♦ 1/3 cup whole blanched almonds
(meat: no, aisle: fruitnuts)

Steps
♦ In a medium-size bowl, beat
margarine, sugar, and eggs until
smooth.
♦ Mix in combined flour, anise seeds,
baking powder, and salt.
♦ Mix in almonds.
♦ Shape dough on greased cookie sheets
into 4 slightly flattened rolls, 1
1/2 inches in diameter.
♦ Bake at 350 degrees until lightly
browned, about 20 minutes.
♦Let stand on wire rack until cool
enough to handle.
♦ Cut bars into 1/2-inch slices.
♦ Arrange slices, cut sides down, on
ungreased cookie sheets.
♦ Bake biscotti at 350 degrees until
toasted on the bottom, 7 to 10 minutes.
♦ Turn and bake until biscotti are
golden on other side and feel almost
dry, 7 to 10 minutes.
♦ Cool on wire racks.
```

The main components are ingredients and steps and some meta-data about the recipe. Ingredients is a collection of individual ingredient components and steps is a collection of individual step components.

Assuming we may not always know the country of origin we can model this collection of recipes as shown in figure 2. Elements are represented by a rectangle: associated attributes are written alongside. The occurrence indicators are the same as for DTD's.

4. RELAX NC

Relax NC is a human-friendly schema language (Fitzgerald, 2002). "A Relax NC schema specifies a pattern for the structure and content of an XML document" (Cowan, Clark, Murata, 2001). There is a java utility, Trang, that will convert the compact form to a machine-friendly xml schema or to the w3 XML schema standard if required. Another java utility, Jing, can be used to validate an XML document against the corresponding Relax NC schema (Fitzgerald, 2002).

Relax NC is based on patterns, it is easy to learn, treats elements and attributes in a similar way and can define groups, collections, occurrences and data types more simply than using a DTD or XML Schema (Cowan et al, 2001; Fitzgerald, 2002). "Relax NG is simply better than either W3C XML Schemas or DTDs in nearly every way" (Mertz, 2003, p.1).

It is possible to "think" in RELAX NC and to build the XML structure **before** marking up a document rather than after. The data analysis could be done using Relax NC and not constructing an elm tree diagram. However I would recommend building the elm tree diagram first as it provides a better visual representation of the data structure and it is easy to convert an elm tree diagram into the corresponding Relax NC schema.

E.g. for the recipe example.

```
start = Recipes
Recipes = element recipes { Recipe+}
Recipe = element recipe
 {RecipeAttributes, Ingredients,
 Steps}
RecipeAttributes = attribute recipeNo
 {xsd:ID},
                       attribute title
 {text},
                    attribute country {text}?,
                    attribute yield {text},
                    attribute calories {xsd:positiveInteger}
```

(RelaxNC is based on patterns. I use a capital letter to start the name of a pattern.)

(RelaxNC supports XML Schema datatypes by default i.e. it "knows" about xsd:positiveInteger).

```
    Ingredients = element ingredients {Ingredient+}
    Ingredient = element ingredient {
          quantity {text},
          attribute meat { "Y"|"N" },
          attribute  aisle {text},
          text
           }


Steps = element steps {Step+}
Step = element step {

        attribute stepNo { xsd:positiveInteger },

        attribute duration {text}?,

        text

        }
```

The corresponding XML for the instance above is shown is Figure 3.

```
<?xml version=”1.0"?>
<recipes>
<recipe recipeNo=”rec143"
        title=”Anise-Almond Biscotti”
        country=”Italy”
        yield=”60 bars”
        calories=”41">
    <ingredients>
      <ingredient quantity=”4 tablespoons” meat=”N” aisle=”dairy”>margarine</
ingredient>
            <ingredient quantity=”3/4 cup” meat=”N” aisle=”baking”>sugar</
ingredient>
        <ingredient quantity=”4" meat=”N” aisle=”dairy”>eggs</ingredient>
       <ingredient quantity=”2 1/2 cups” meat=”N” aisle=”baking”>all-purpose
flour</ingredient>
         <ingredient quantity=”2 teaspoons” meat=”N” aisle=”spices”>crushed
anise seeds</ingredient>
       <ingredient quantity=”1 1/2 teaspoons” meat=”N” aisle=”baking”>baking
powder</ingredient>
         <ingredient quantity=”1/4 teaspoon” meat=”N” aisle=”spices”>salt</
ingredient>
      <ingredient quantity=”1/3 cup” meat=”N” aisle=”fruitnuts”>whole blanched
almonds</ingredient>
    </ingredients>

    <steps>
         <step stepNo=”1">In a medium-size bowl, beat margarine, sugar, and
eggs until smooth.</step>
       <step stepNo=”2">Mix in   combined flour, anise seeds, baking powder,
and salt.</step>
        <step stepNo=”3">Mix in almonds.</step>
       <step stepNo=”4">Shape dough on greased cookie sheets into 4 slightly
flattened rolls, 1 1/2 inches in diameter.</step>
         <step stepNo=”5" duration=”about 20mins”>Bake at 350 degrees until
lightly browned.</step>
       <step stepNo=”6">Let stand on wire rack until cool enough to handle.</
step>
        <step stepNo=”7">Cut bars into 1/2-inch slices.</step>
        <step stepNo=”8">Arrange slices, cut sides down, on ungreased cookie
sheets</step>
          <step stepNo=”9" duration=”7 to 10 minutes”>Bake biscotti at 350
degrees until toasted on the bottom.</step>
          <step stepNo=”10" duration=”7 to 10 minutes”>Turn and bake until
biscotti are golden on other side and feel almost dry.</step>
        <step stepNo=”11">Cool on wire racks.</step>
    </steps>
</recipe>
</recipes>
```

Figure 3: XML for recipe example.

5. SUMMARY OF STRUCTURE

The art of XML design can be summarised as:

A Identify the components and their attributes. The listing of components may or may not follow the structure of the original document. It is the data and the data structures that are important.

B Identify the structure. Build a shrub structure. Identify groups of components and look for collections. Often there will be a header group containing meta-data (XML Patterns).

C Model the structure. Build an elm tree diagram then use this to write the schema using RELAX NC.

6. DISCUSSION – A TALE OF TWO CLASSES

A semester 2, 2002 class was taught XML by following the early chapters of "Learning XML" (Ray, 2001). Students were taught mark-up concepts and the rules for well-formed documents. There was some discussion of elements and attributes and examples of collections and groups. The first stage of their assignment was to mark-up a course outline. There was a wide range of solutions. All were well-formed but some were better than others. Most students used only elements. Many included presentation aspects, e.g. footnotes and section headings. They then produced a corresponding DTD. The fact that tools such as XMLSpy can do this automatically shows that it is not a difficult task. It is simply a reflection of the XML mark-up used in the document. The third stage was to write xsl that would extract the data from the XML document and reproduce the original course outline in XHTML format (These students have previous programming experience and are familiar with html). It was when writing the xsl code that they realised that changing the structure of the XML document could have a major influence on the ease of subsequent processing. Many students revisited their XML structure.

For the January 2003 summer school class, the teaching approach used was as outlined in this paper. From the teaching perspective it seemed a more logical way to proceed and the students appeared to better understand the concepts and the issues. They all produced "good" XML structures and efficient xsl stylesheets.

Students were asked to record the time taken for the stages of the assignment and to comment on the process. A comparison of the times taken to complete equivalent components suggests that the differences are significant at the 95% confidence level. However this only adds to the anecdotal evidence that using an elm tree diagram and Relax NC is a better methodology for XML mark-up. The two classes were self-selected and not a random selection from the same student population.

6.1 Comments from the 2002 class (traditional XML and DTD)

"As I progressed through the assignment, I found myself modifying my XML."
"I had to re-design the XML in order to make life a little bit easier."
"The task which took the most time was coding the document in XML. This was hard because I needed to make decisions about how to structure the information which were difficult because I didn't know what sort of issues we would be facing later on."
"I found that while doing the xslt I changed quite a lot of the structure in the XML."
"The structure of my XML did not allow me to use xsl's looping to maximum effect."
"I had to change my XML and DTD when I created the XSLT."
"Next time I would spend more time on the design of the XML."

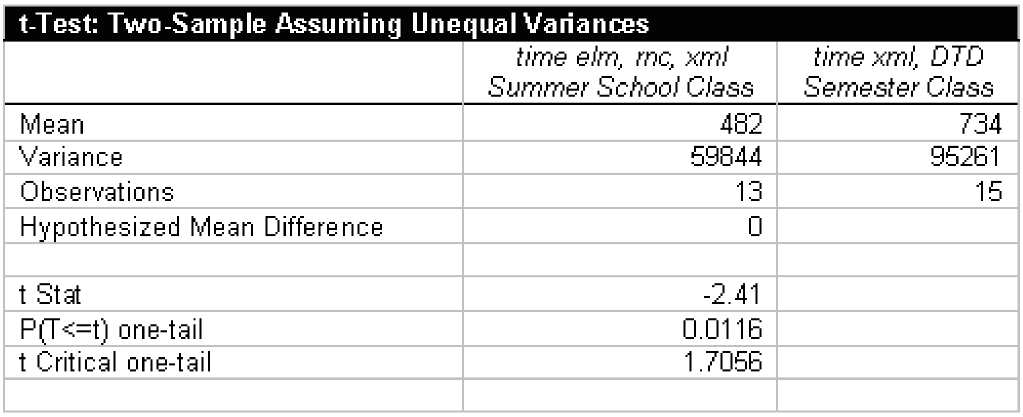
6.2 Comments from the 2003 class (XML and Relax NG)

"Next time I would spend more time and care on the elm tree diagram – its importance is obvious now."
"Time spent on analysis paid off."
"The best thing I found was doing the elm tree diagram."
"RNC easier and quicker than writing a DTD." (from someone who had previously attempted this paper).
"The elm tree diagram was a vital base for the rest of the assignment."

7. CONCLUSIONS

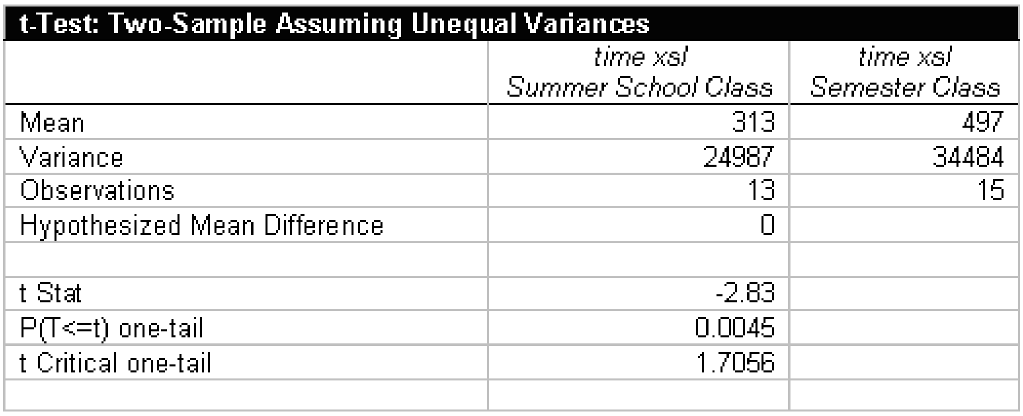
XML mark-up is a mark-up of the data structure of a document. It makes sense to do the data analysis first. Elm tree diagrams were developed for the data analysis of documents and can be used in conjunction with best practice rules such as identifying collections, creating groups, using attributes for element meta-data, and using an xml pattern where appropriate. Relax NC is not only a more human-friendly schema language than DTD or XML Schema it is also more powerful and flexible, e.g. its ability to specify interleaved elements within a group (Mertz, 2003). Using the above methodology for teaching XML mark-up appears to be better from both the teaching and

Table 1. Comparison of time taken to create the XML and corresponding DTD or Relax NC Schema



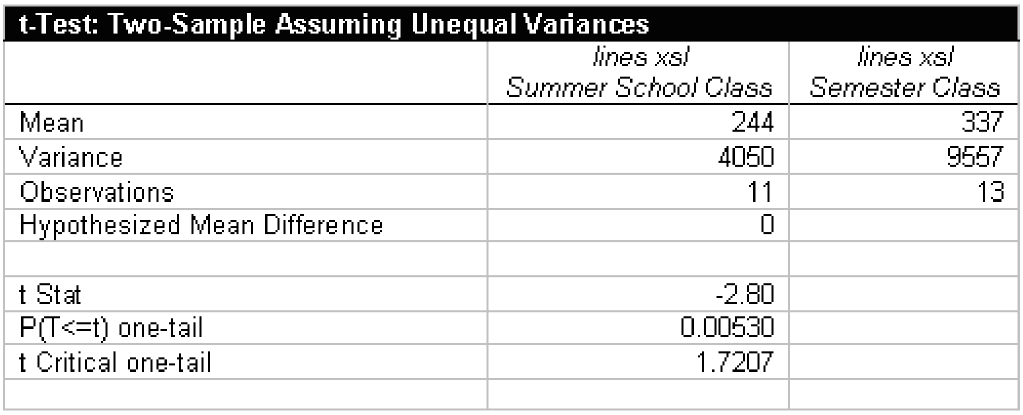
t-Test: Two-Sample Assuming Unequal Variances		
	time elm, rnc, xml *Summer School Class*	*time xml, DTD* *Semester Class*
Mean	482	734
Variance	59844	95261
Observations	13	15
Hypothesized Mean Difference	0	
t Stat	-2.41	
P(T<=t) one-tail	0.0116	
t Critical one-tail	1.7056	

Table 2. Comparison of time taken to write the xsl code



t-Test: Two-Sample Assuming Unequal Variances		
	time xsl *Summer School Class*	*time xsl* *Semester Class*
Mean	313	497
Variance	24987	34484
Observations	13	15
Hypothesized Mean Difference	0	
t Stat	-2.83	
P(T<=t) one-tail	0.0045	
t Critical one-tail	1.7056	

Table 3. Comparison of lines of xsl code



t-Test: Two-Sample Assuming Unequal Variances		
	lines xsl *Summer School Class*	*lines xsl* *Semester Class*
Mean	244	337
Variance	4050	9557
Observations	11	13
Hypothesized Mean Difference	0	
t Stat	-2.80	
P(T<=t) one-tail	0.00530	
t Critical one-tail	1.7207	

the student point of view. The students appreciated the more directed approach and found elm tree diagrams and Relax NC easy to learn and use which was reflected in the time they spent on the assignment and in the quality of the XML mark-up and code they produced.

REFERENCES

Baker M. (2001). “Designing XML Tagging languages”. XML Journal Vol 2, 8 pgs 38-42.

Brandin C (2003) “Maximising the Usefulness of XML”. XML Journal Vol 3, 12 pgs 16-19.

Carlson D. (2001) “Modelling XML Applications with UML” Addison-Wesley.

"Categorized XML Patterns" Accessed September 2002 www.xmlpatterns.com/categoryAllPatterns.shtml

Cowan J & Murata M. (2001) "Relax NG Specification: Committee Specification 3 December 2001" Accessed February 2003 http://www.oasis-open.org/committees/relax-ng/spec.html

Cowan J., Clark J., Murata M. (2001) "RELAX NG Tutorial Compact Syntax Committee Specification 3 December 2001" Accessed September 2002 http://home.ccil.org/~cowan/XML/compact-tutorial-20011203.html

Daconta M. C. (2001). "Are Elements and Attributes Interchangeable?". XML Journal Vol 2, 7 pgs 42-44.

"Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation 6 October 2000". Accessed May 2001. <http://www.w3.org/TR/REC-xml>

Fitzgerald M. (2002) "RELAX NG's Compact Syntax". Accessed September 2002: http://www.xml.com/lpt/a/2002/06/19/rng-compact.html

Harold E. R. & Means W. S. (2001) "XML in Nutshell". O'Reilly

Hoenisch S. (2001). "Structuring Documents with XML". XML Journal Vol 2, 6 pgs 40-43.

Jelliffe R. (1998) "The XML and SGML Cookbook: recipes for structured information". Prentice Hall PTR.

Maler E. & el Andaloussi J. (1996) "Developing SGML DTDs: from Text to Model to Markup". Prentice Hall PTR.

Mertz D. (2003) "XML Matters: Kicking Back with RELAX NG Part 1" Accessed March 2003

http://www-106.ibm.com/developerworks/xml/library/x-matters25.html?dwzone=xml

Morrison M. et al (2000) "XML Unleashed". Sams.

Muench S. (2002). "The Attribute/Text Conundrum". Accessed November, 2002: www.xmleverywhere.com/newsletters/20000525.htm

Patrizio A. (2002) "XML returns to its authoring roots". XML & Web Services, vol 3, 4.

Ray E. T. (2001) "Learning XML 1st edition". O'Reilly.

"RNC Distilled" (2002). Accessed October 2002: http://xmldistilled.com/tutorials/free/slides/01.html

St Laurent S. (1998) "Why XML?". Accessed Novmber 2002. <http://www.simonstl.com/aricles/whyxml.htm>

St Laurent S. (2002) "An ascetic view of XML best practices". Accessed November 2002. <http://monasticxml.org/index.html>

W3C Communications Team (2001). "XML in 10 points" Accessed November 2002 www.w3.org/XML/1999/XML-in-10-points.html.en

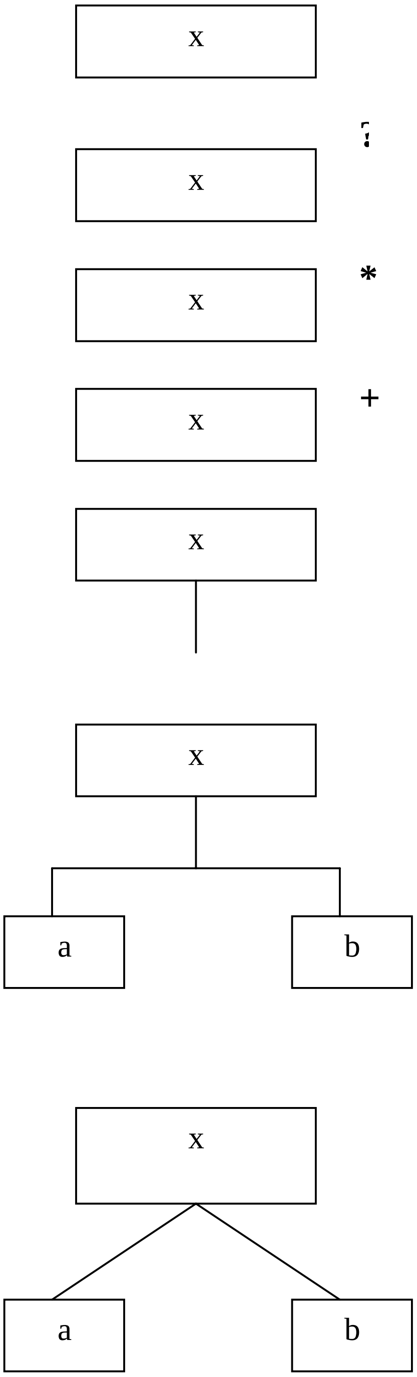
"XML Attributes". Accessed June 2000: www.w3schools.com/xml/xml_attributes.asp

"XML Tutorial" Accessed March 2003

www.w3schools.com/xml/default.asp

APPENDIX Elm Tree Diagram Notation Summary

Figure 4. Elm tree diagram notation summary.



Element x is required

Element x is optional

Element x is optional and repeatable

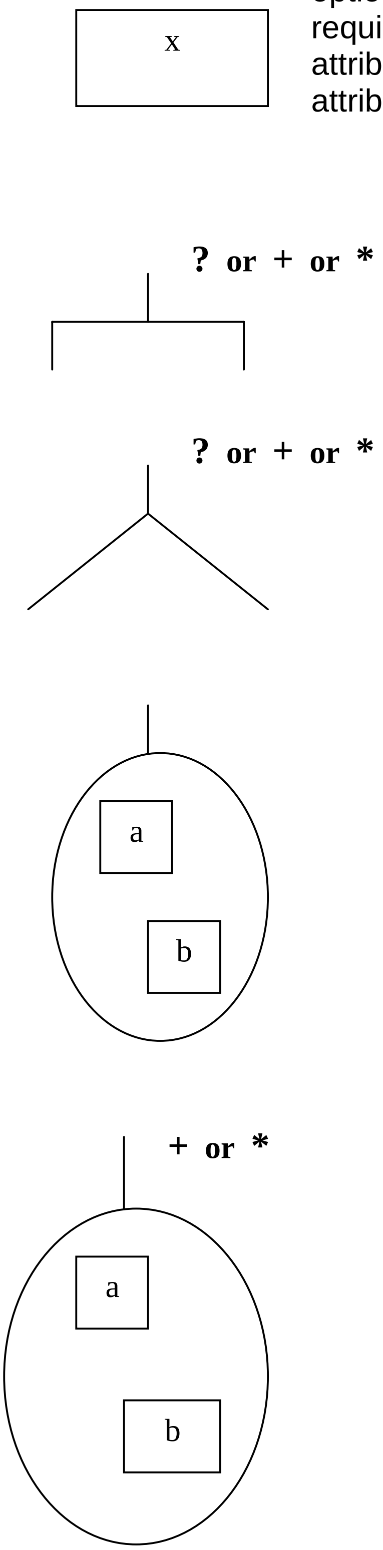
Element x is required and repeatable

Element x has the content model …

Element x contains element a followed by element b

Element x contains either element a or element b

Figure 5. elm tree diagram summary continued.



optional attribute
required attribute
attribute with token default=(a|b|c)
attribute with string default (“d”)

Sequential group is optional, optional-repeatable, or required-repeatable

Either-or group is optional, optional-repeatable, or required-repeatable

Element a and element b must appear, and can be in any order

Element a and element b can appear repeatedly or not at all, in any order (**+** means at least one element is required)