Wright State University

CORE Scholar

Kno.e.sis Publications

The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis)

8-2008

An XML-Based Approach to Handling Tables in Documents

Krishnaprasad Thirunarayan
Wright State University - Main Campus, t.k.prasad@wright.edu

Trivikram Immaneni

Follow this and additional works at: https://corescholar.libraries.wright.edu/knoesis

 Part of the Bioinformatics Commons, Communication Technology and New Media Commons, Databases and Information Systems Commons, OS and Networks Commons, and the Science and Technology Studies Commons

Repository Citation

Thirunarayan, K., & Immaneni, T. (2008). An XML-Based Approach to Handling Tables in Documents. *Journal of Intelligent Systems, 17* (1-3), 215-228.
https://corescholar.libraries.wright.edu/knoesis/907

This Article is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.

An XML-Based Approach to Handling Tables in Documents

Krishnaprasad Thirunarayan and Trivikram Immaneni

Department of Computer Science and Engineering,
Wright State University, Dayton, OH 45435, USA

ABSTRACT

We explore the application of Extensible Markup Language (XML) technology for handling tables in legacy semi-structured documents. Specifically, we analyze annotating heterogeneous documents containing tables to obtain a formalized XML Master document that improves traceability (hence easing verification and update) and enables manipulation using XSLT stylesheets. This approach is usefol when table instances fer outnumber distinct table types because the effort required to annotate a table instance is relatively less compared to formalizing table processing that respects table's semantics. This work is also relevant for authoring new documents with tables that should be accessible to both humans and machines.

KEYWORDS

data management/language, methodology, techniques, tools; table annotation/ manipulation; XMUXSLT application; semi-structured data representation for traceability

I. INTRODUCTION AND MOTIVATJON

Extensible Markup Language (XML) has been used to annotate documents with metadata in the realm of document processing and content extraction

Correspondence to: K. Thirunarayan. e-mail: {t.k.prasad, immaneni.2}@wright.edu

to be read and maintained by humans. XML has also been widely used as a standard text-based format for information exchange/ serialization in the context of Web services for manipulation by machines. These two prevailing applications of XML have been respectively termed *document-centric* and *data-centric*. This paper explores an approach to unifying these two views by using XML elements to materialize abstrac syntax, and together with XML attributes, to represent the semantics via annotation, obtaining an XML Master document that is both machine processable and can serve as a basis for human sensible presentation. This work can be beneficial for formalizing, representing, and manipulating (domain-specific) content in legacy semi-structured documents, and for authoring new documents that are simultaneously human and machine consumable.

To better situate the current work, we consider the relationship between information extraction and document authoring for the Semantic Web, in terms of client-server paradigm. Information Extraction deals with automatic filtering of legacy documents and filling-in a pre-specified, domain-specific template by a client/end user. In contrast, Semantic Web requires authoring of documents conforming to a fixed ontology by a server/document creator (Antoniou & van Harmelen 2004, Fensel et al 2003, W3C).

This paper pursues a pragmatic, semi-automatic approach to annotation that straddles these two extremes. The ultimate goal is to develop the document and its formalization hand in hand and keep them side by side to improve trace-ability by maintaining an implicit link between original document fragments and its formalization. The documents of interest are heterogeneous and semi-structured, containing text and tables. For example, consider the following table type shown in Table I that appears frequently in materials and process specs, which gives tensile strength and yield strength as a function of the thickness of a specimen:

Handling tables requires recognizing table layout and understanding table content, for subsequent manipulation. The table layout alone cannot be used to understand tables automatically because the semantics of a table is normally gleaned by a user from the heading labels and captions relating various columns and rows. A potential semi-automatic approach to dealing with tables is to develop a catalog of table types and its processing via XML and XSLT style-sheets and then make explicit the interpretation of each table instance occurring

TABLE 1

Tensile data table

Thickness {in)	Tensile Strength {ksi)	Yield Strength @0.2% offset (ksi)
0.5 and under	165	155
0.50 - 1.00	160	150
1.00 - 1.50	155	145

in a document manually, using specific XML-based annotations. The composite XML document not only provides a basis for human sensible view but also supports direct machine manipulation. The intertwining of the annotations formalizing the content with the original text in the XML document promotes traceability, hence easing verification and update. Furthermore, given that formalizing the document content completely is, in general, impossible and impractical, it is usef ul to be able to fall back on the related pieces of original source text for additional information or to provide context for the generated annotations. Overall, this approach holds promise, in so far as the number of table instances far outnumbers the number of table types because the manual effort required to annotate a table instance is relatively less compared to formalizing table processing that respects the table's semantics. The cluttering effect of XML tags on readability can be minimized by displaying suitable views of the XML source for editing purposes, along the lines of what HTML editors do.

In Section 2, we discuss several interesting issues and techniques related to table handling in the literature. Specifically, we situate, analyze, and review our past work on formalizing and querying tables in Water (Thirunarayan 2005). In Section 3, we present details on formalizing tables in XML and using XSLT stylesheets for table manipulation, and discuss obstacles, advantages, and disadvantages. In particular, we illustrate how annotations can be embedded into table text, and the resulting annotated document can be made XML compliant using a simple, mechanizable transformation. In Section 4, we conclude with a summary of remaining problems to be solved and suggestions for future work.

2. RELATED WORK

There are a number of practical, orthogonal research issues pertinent to handling of tables found in documents as discussed below:

- Extraction of Tabular Data: Tables in documents that are available in plain text, MS Word, or PDF fam may be hand-formatted or created using table primitives .The techniques we have developed in the past can be used to convert MS Word document into plain text that delimits and preserves table layout, so that it is human readable (Thirunarayan et al 2005). In particular, we used this approach to pre-process legacy materials and process specs from GE-AE (General Electric - Aircraft Engines). The work on table extraction (Pinto et al 2003) deals with the isolation of complex tables from the rest of the text, and identifying the title, row headings, cell boundaries, etc. Similarly, earlier work on Wrapper Induction and its robust generalization to accommodate visual cues implicit in the geometry of the tables can assist in table extraction from HTML documents (Kushmerick 2000; Cohen 2002). In the realm of content extraction from specs, we also need to deal with complex column headings. Similarly, there are systems that address issues such as the recognition of table components in a text document (e.g., TINTIN (Pyreddy et al, 1997; Pande, 2002; Zanibbi et al 2004) or the repre-sentation of structure and flexible presentation of tables, e.g., Tabula Magica (Silberhorn 2001). For the current state of the art, to expect the extractor to manually identify and tag the table components and focus on how to interpret the domain-specific table content seems reasonable.
- Representation of Tabular Data for Semi-automatic Translation : Several different issues must be considered regarding the semantics of tabular data:

  Consider an XML-inspired approach to providing semantics to tables in plain text that promotes traceability , where a table contains both the headings and the data. The precise relations among the various values in a row/column are tacit in the heading labels, and obvious to extractors. However, this semantics needs to be made explicit to do any machine processing, but storing only a semantics rich translation in

a new fonnalism is not always conducive to human comprehension or flexible manipulation. So, the representation language should have the provision to more or less preserve the grid layout of a table to promote readability and enable changes to the original table to be easily incorporated in text, while describing the interpretation of each row/column in a way that is flexible and applicable to all rows/columns for further machine manipulation. We have looked into two different avenues, each with its own pros and cons (Thirunarayan, 2005). In Water (Plusch, 2003), annotatioo definition can encapsulate interpretation and be treated as a method, while the annotated data can be viewed as a method call. (See Section 3 for a concrete example.) This novel view of annotation enables the interpretation of data to be described in an additive fashion, shared among multiple annotated table instances of the same kind Unfortunately, this Water program is not a well-formed XML document, thereby losing the ability to reuse techniques and tools developed for manipulating XML documents. Furthermore, Water is not conducive to convenient embedding of the formaliz.ation into the original document because Water requires the original text to be delimited and incorporated as comments. On the other hand, a well-formed XML annotated table that tags each table cell intersperses tags with table data, which is not always desirable considering the effort required to create it and the resulting "ugly" form.

- Another approach worthy of exploration is to define a language of table expressions with compositional semantics that enables one to build and manipulate tables with headings algebraically (Wolfram, 2003).
- At this juncture, a viable approach to dealing with tabular information is to develop a catalog of predefined tables and map the tabular data into a set of pre-defined tables, possibly qualified. For instance, a complex table can be built as a union of qualified simple tables. Overall, manual mapping of complex tables into simpler regular structures have the following benefits:
  - It provides semantics to data, thereby removing any lurking ambiguities.
  - It provides natural expression of data for traceability and ease of use.

  - It enables automatic manipulation, that is, for querying and translation.
- Manipulation of Tabular Data: Once the problem of table representation is solved, we need to develop the corresponding language and techniques for querying, combining and detecting conflicts among related tables. For instance, TANGO (Tijerino et al 2003) use table-equivalent data to generate ontologies. Specifically, they define what constitutes a table, how one can recognize a table and infer the table schema using background information provided by WordNet, tokenizers, and data type descriptions (data frames), and then show how to combine schemas obtained from related tables to build ontologies.

3. AN XMUXSLT-BASED APPROACH TO TABLES

Recall that heterogeneous, semi-structured text documents are not conducive to machine processing, so it makes sense to develop techniques to abstract, formalize, and represent their content in a more structured manner. In order to ascertain the soundness of the formalization/translation, it is important to link the original document fragments with their formalization. The additional data structures needed to capture this association can be simplified if the formalization can in fact be *embedded* in the original document. Furthermore, the composite document has potential to be readily understood and updated by a human user due to its resemblance to the original document. The document-centric and data-centric views of XML seem to provide a means to the desired end:

- XML can encode text and tabular data, to make explicit abstract syntax and the semantics via predefined, domain specific annotations, and
- XSLT stylesheets can be used to describe various interpretations respecting the semantics for formal manipulation, in a modular fashion.

Relationships described in plain text can be formalized using XML elements and XML attributes. Dealing with tables, however, is much harder, as discussed below.

Water, an XML-inspired programming language, provides a rich substrate for formalizing and querying heterogeneous documents (Thirunarayan, 2005). The annotated data can be *interpreted* as a method call, and the XML-element as a method, as illustrated below in the context of the example in Table 1.

```
<!- Thickness (mm) Tensile Strength \ksi) Yield Strength (ksi) ->
tableObj .<setHeading thickness strength.tensile strength.yield/>
<!-            0.50 and under          165               155 ->
tableObj .<addRow 0.50                 165               155 /> ...
```

Each table row is annotated using a tag to get an XML element that becomes a method invocation on a table object in Water. If the rows require dissimilar treatment, then different tags can be used. Each type used in the table is defined using Water's def class construct. The table class supports methods that manipulate the content as intended by its semantics.

```
<defclass thickness value=required=number units="mm"/>
<defclass strength value=required=<nunber units="ksi">
        <defclass tensile/>
        <defclass yield offset="0.2/>
</defclass>
<defclass table rows=required=vector heading=optional=vector>
        <defmethod setHeading t ts ys>

        </>
        <defmethod addRow smin smax ts ys>

        </>
        <defmethod  computeYieldStrenqth>

        </>
        <defmethod computeTensileStrength>

        </>
</>
```

Ideally, the tabular data in each document is only annotated, while factoring out annotation definitions separately as domain-specific background knowledge. Note, however, that the correspondence between formal parameters and actual arguments is positional (as seen in the above Water code),

yielding an ill-formed XML document in the presence of tables. Specifically, this approach does not permit flexible embedding of annotations into text, or use of XML techniques and tools such as XSLT, because the annotated document violates XML syntax.

We will now attempt to annomte a document containing the table text, to capture its semantics via suitably chosen XML tags and XSLT stylesheets that manipulate the table according to its semantics. Any deviation from XML well-formedness criteria will be remedied by *reinterpreting* the resulting document in terms of an "equivalent" XML document. Specifically, we will reinterpret the positional association of actual arguments to formal parameters using fixed name-based associations that can be captured in XML using attribute-value pairs and manipulated using XSLT stylesheets. That is, the call <mthd "al" "a2" "a3" ...> will be turned into <mthd one="al" two-"a2" three="a3" . ..> to conform to XML syntax. Once this association is clarified, the annotated data can in fact be interpreted differently by programming-in different interpretations for the XML-element using different XSLT stylesheets. For instance, one can recover just the text sans the annotations, verify integrity constraints, transform data, or even facilitate data querying (such as by mapping the annotated document into Prolog-like syntax). This mechanism also enables incorporation of common-sense knowledge and domain-specific background information and checks, in a modular fashion. Concretely, the requirement that temperature must be a number (static type), or should be in the range from 300°F to 500°F (dynamic constraint) can be made explicit by defining temperature constraints via XSLT stylesheets.

TABLE 2

Spec tensile data table as text

Thickness (in)	Tensile Strength (ksi)	Yield Strength @0.2% offset (ksi)
O.S and under	165	*ISS*
O.SO - 1.00	160	ISO
1.00 - *I.SO*	*ISS*	145

We illustrate the XML-based representational issues and XSLT-based transformational details using an illustrative example of a tensile data table taken from materials and process specs shown in Table 1. This table can be extracted as text from an MSWord document as shown in Table 2 and subsequently, manually annotated to bring out its structure as follows:

```
<table type="Tensile">
<dependency pararneter="Yield Strength
                          narne "ield Offset" value="0.2\/>
<tableSchema
            "Thickness(rnin)" "Thickness(max)" "Tensile Strength"
"Yield Strength"/>
<tableUnits "inch"     "inch" "ksi"   "ksi" I>
<tableData   "0"            "0.50"   "165"   "155" />
<tableData  "0.50"     "l.00"  "160"   "150" />
<tableData  "l.00"     "l.50" "155"   "145" I>

</table>
```

In particular, the double quotes and annotations clearly delimit atomic values, relate data, and clarify the column headings, the units of measure, and other dependencies that are crucial for proper interpretation of the tensile table data. This annotation process can be facilitated by a tool that manages XML Master document and provides a palatable view of it for editing purposes, such as via colored view of the tagged information based on its type (Cunningham, 2002; Tijerino, 2003). Unfortunately, the annotated table is not a well-formed XML fragment. To ensure well-formedness, we have to come up with a simple, natural, regular scheme for automatically deriving an equivalent XML document, for example, by introducing sequencing attributes one, two, three , ..., etc to capture positional associations via named-associations as follows.

```
<?xml version="l.0" encoding="IS0-8859-1"?>
<table type="Tensile">
<dependency pararneter="Yield Strength" narne="Yield Offset"
                                                    value="0.2\">
<tableSchema  one="Thickness(rnin)" two-"Thickness(max)"
                  three="Tensile Strength" four="Yield Strength"/>
<tableUnits  one="in"   two="in"   three="ksi"  four="ksi"  />
<tableData   one-"0"   two="0.50"  three-"165"  four="l55"  />
<tableData   one="0.50"  two="l.00"  three="l60"  four="l50"  />
<tableData   one="l.00" two="l.50" three="l55" four="l45" />

</table>
```

This *in-place* formalization of a table instance, when augmented with reusable, table type specific XSLT stylesheets yields a representation that exhibits a prescribed semantics and is both machine manipulable and can be made human accessible. In particular, XSLT stylesheets can be designed to carry out the following operations on the XML document that contains both the original table text and the annotated table:

- *Query:* to perform table look-ups.
- *Transform:* to change units of measure such as from standard SI units (International System of units) to FPS units (Foot, Pound and Second system of units) and vice versa.
- *Format:* to display the table in HTML form.
- *Extract:* to recover the original table in text form.
- *VerifY:* to check static semantic constraints on table data values.

We now present illustrative examples of the transformations that can be carried out using XSLT stylesheets.

- Given a thickness, determine the tensile strength or the yield strength at yield offset of 0.2%.

The stylesheet ignores text data, determines the appropriate tensile table formalization in XML, and then searches through this table to determine the applicable strength value. For the example tensile table in XML, for the thickness value of 0.25 inch, the looked up tensile strength value is 165 ksi (kilo-pounds per square inch) and yield strength value at 0.2% offset is 155 ksi.

- Given the table in FPS units, create an HTML table that displays the data in both FPS units and SI units.

For the example tensile table in XML that has thickness values in inch and strength values in ksi, we can generate HTML table for presentation that shows thickness values in both inch and mm (millimeter), and strength values in both ksi and MPa (MegaPascal or Newton per square millimeter) as shown in Table 3.

TABLE 3

Generated tensile data presentation

Thickness (min)	Thickness (max)	Tensile Strength	Yield Strength @0.2% Yield Offset
inch(mm)	inch(mm)	ksi(MPa)	ksi(MPa)
0(0)	0.5(12.7)	165(1137.6)	155(1068.7)
0.50(12.7)	1.00(25.4)	160(1103.2)	I 50(1034.2)
1.00(25.4)	1.50(38.1)	155(1068.7)	145(999.7)

TABLE4

Chemical composition data for UNS R55l 1I (The Navy Alloy)

Element	Composition (min)	Composition (max)
Al	4.5%	5.5%
Sn(Zr,V)	0.6%	1.4%
C	0	0.08%
Ti		Balance

To appreciate the need for customized treatment of tables, consider that one can have superficially similar tables that mean different things based on the context. For instance, the interpretation of thickness depends on the cross· section of the product. For square (resp. hexagonal, circular) cross-section, the thickness corresponds to length (resp. the distance between parallel faces, diameter). The strength values depend on the direction of loading -whether it is short transverse, long transverse, or logitudinal. The tensile strength table, and chemical composition (chemistry) table associate value ranges with primitive values. In a tensile table, thickness determines the tensile strength, whereas in a chemical composition table, it is the chemical element whose composition is constrained by the range values, as shown in Table 4. The composition can itself be given by weight or by volume; it may specify the ingot chemistry or the product chemistry.

To ensure practicality of this approach, we need a tool (1) with MS FrontPage like interface, for which the Master document is the annotated form and the user explicitly interacts/edits via a convenient view of the annotated document, and (2) with support for *export as XML* option, which can tum the annotated document into a well-formed XML document. For instance, in the current context, this means adding attributes such as one, two, three, ..., etc using a transformation shown below that can be carried out only outside of XML/XSLT:

<elem "Pl" "P2" "PJ" .• .> ""><elem one•"Pl" two•"P2" three•"PJ" ...>.

Ideally, we do not need to create a separate annotated table, distinct from what can be obtained by annotating the original document, which further provides the context for interpretation of data and is amenable to track revisions to the embedded tables.

4. CONCLUSIONS AND FUTURE WORK

We have explored techniques to imbue table instances with machine-processable annotations to obtain an XML Master document, to promote traceability. The positional association of table data with table headings is captured via semi-automatically created named associations to obtain an XML-compliant document that enables reuse of XML techniques and tools (such as XSLT) for flexible interpretation and manipulation of text documents containing tables. One can also view this approach as mapping a larger collection of annotated documents into "equivalent" XML documents, or reinterpreting an annotated fragment such as <elem "Pl" "P2" "P3" • ..*I* > as a concise description of sequentially assigned attributes in an XML fragment such as <elem one=-"Pl" two=-"P2" three=-"P3" .../>.

Encoding tables in Prolog for querying provides a flexible alternative to using Water or XML/XSLT. However, for document representation and manipulation, an XML/XSLT-based approach is more powerful. Furthermore, the treatment of more general tables, such as those containing multiple columns with common headings or containing non-uniform rows combining multiple tables or containing tables multiple units of measure, etc is non-trivial.

The essence of Semantic Web is to make explicit semantics of data in a machine processable form. What we have accomplished here is a pragmatic first step in the context of tables that enables programming in various interpretations respecting the semantics of an annotated table.

REFERENCES

Antoniou, G., and van Harmelen, F. 2004. A semantic web primer. Cambridge, MA, USA, The MIT Press.

Cohen, W.W., Hurst, M., and Jensen, L.S. 2002. A flexible learning system for wrapping tables and lists in HTML documents, In: *The Proceedings of the Eleventh International World Wide Web Conference (WWW 2002)*, 232-41.

Cunningham, H., Maynard, D., Bontcheva, K., and Tabian, V. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications, In: *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL '02).*

Fensel, D., Hendler, J., Lieberman, H., and Wahlster, W., Editors. 2003. *Spinning the semantic web: Bringing the world wide web to its full potential*. Cambridge, MA, USA, The MIT Press.

Kushmerick, N. 2000. Wrapper induction: Efficiency and expressiveness, In: *Artificial Intelligence*, 118, 15-68. (Special issue on Intelligent Internet Systems.)

Pande, A.K. 2002. *Table understanding for information retrieval*, Master's Thesis, Virginia Polytechnic Institute and State University, 55 pages.

Pinto, D., Mccallum, A., Wei, X., and Croft. W.B. 2003. Table extraction using conditional random field In: *SIGIR '03 Conference*, 235-42.

Plusch, M. 2003. *Water: simplified web services and XML programming*, Wiley Publishing. (http://www.waterJanguage.org!)

Pyreddy, P. and Croft, W.B. 1997. TINTIN: A system for retrieval in text tables, *2nd ACM International Conference on Digital 'Libraries*, 193-200.

Silberhorn, H. 2001. TabulaMagica: An integrated approach to manage complex tables, In: *ACM Symposium on Document engineering*, 68-75. W3C Semantic Web URL: http://www.semanticweb.org/.

Tijerino, Y.A., Embley, D.W., Lonsdale, D.W., and Nagy, G. 2003. Ontology generation from tables, In: *Proceedings of the 4th International Conference on Web Information Systems Engineering*, 242-9.

Thirunarayan, K., Berkovich, A., and Sokol, D. 2005. An information extraction approach to reorganizing and summarizing specifications, In: *Information and Software Technology Journal* 47(4), 215-32.

Thirunarayan, K. 2005. On embedding machine-processable semantics into documents, In: *IEEE Knowledge and Data Engineering Journal* 17 (7), 1014-8.

Wolfram, K. 2003. *Compositional syntax and semantics of tables,* SQRL Report No. 15, Dept. of Computing and Software, McMster University, Hamilton, Ontario, Canada, 1-62.

Zanibbi, R., Blostein, D., and Cordy, J. R. 2004. A survey of table recognition: models, observations, transformations, and inferences. *International Journal of Document Analysis and Recngnition,* 7 (I), 1-16.