Encoding and Presenting Interlinear Text Using XML Technologies

Baden Hughes, Steven Bird and Catherine Bow
Department of Computer Science and Software Engineering
University of Melbourne
Victoria 3010, Australia
{badenh, sb, cbow}@cs.mu.oz.au

Abstract

Interlinear text is a common presentational format for linguistic information, and its creation and management have been greatly facilitated by the development of specialised software. In earlier work we developed a four-level model and corresponding formal specification for interlinear text. Here we describe a suitable XML representation for the model and show how it can be rendered into a variety of convenient presentational formats. We conclude by discussing architectural extensions, an application programming interface for interlinear text, and prospects for embedding the interlinear model into existing applications.

1 Introduction

Interlinear text is a standard presentational form for displaying a source text aligned with variety of linguistic annotation phonological, morphological and syntactic analyses, glosses and translations. An example of Yidinj interlinear text is shown in Figure 1.

Interlinear texts can vary in the number of rows, the content and level of analysis for each row, and certain aspects of layout. However, as we survey a broad range of cases, we can observe consistent patterns in layout. These patterns have lead us to propose an abstract model of interlinear text which is sufficiently general that it encompasses the majority of cases based on a survey of print and electronic representations of interlinear text (Bow, Hughes and Bird 2003). As we will show here, the model has a natural representation in XML, which can be used to generate a variety of useful visualisations. We begin by describing the model in §2, then propose a suitable XML representation (§3). In §4 we discuss common presentation styles, and in §5 we show how they can be implemented using XSL. Finally, we report our work on a prototype (§6) and discuss future research (§7).

njundu	*wanjdjam*
you-SA	where-ABL
Where have you come from ?	

Figure 1: Yidinj Interlinear Text

2 The EMELD Interlinear Text Model

For the purposes of this discussion we adopt the following nomenclature. By *interlinear text* we mean a written record of an external linguistic event, consisting of a transcription aligned with linguistic analysis. A *line* of interlinear text refers to a row of transcription plus all the rows of analysis pertaining to it. Within the line, there is a *horizontal modality* of words and their analysis, plus there is the *vertical modality* of row elements to indicate the structure of the interlinear text. Additionally there is the *vertical alignment* of one row above the other; typically there is consistency from one block of rows to the next in the vertical alignment of these rows.

Within the context of the EMELD project (EMELD, 2000), Bow, Hughes and Bird (2003) proposed a four-level, hierarchical model of interlinear text consisting of Text, Phrase, Word and Morpheme levels. A *phrase* is a collection of transcription and its interlinear analysis arrayed across two or more rows, normally represented in interlinear text as beginning on a new line, and wrapping only if necessary. A *word* is a smaller collection of material (e.g. transcriptions, morphemes and glosses) that must be kept together on the same line. A *morpheme* is the smallest possible level of alignment between linguistic forms and their meanings. In contrast, a *text* is the highest level of structure corresponding to the external linguistic artefact being investigated. Just as it is possible for a word to contain a single morpheme, a phrase may contain a single word, and a text may contain a single phrase. Thus there is no prior commitment to the total length of this external linguistic artefact. According to this four-level scheme, the user has flexibility in the assignment of content to levels, and the decision may be influenced by both linguistic considerations (what is a ‘word’? Is this one text or two?) and layout considerations (what should be kept on the same line in a display?). Additionally, at any particular level, analysis may be optionally omitted.

In this model, two rows are represented within a single node of the hierarchy if they are aligned with each other. For example, morphemes and their glosses, while being displayed on two different rows, are both represented at the Morpheme level of the hierarchy, given their 1:1 correspondence. The hierarchical model allows for the concatenation of morphemes into words, words into phrases, and phrases into text, as required by the interlinear text. It also allows for complex information structures to be represented in simple tree diagrams.

3 The XML Representation

We now turn to a discussion of the XML representation of the EMELD model. In this representation, the textual material has a number of levels, which can be considered nodes in an XML document. Each level consists of some content, and a sequence of children at subsequent levels. The content is given a user-defined type which documents its intended semantic interpretation.

```
<interlinear-text>
  <item type="user-defined">
    Content at the text level, such
    as metadata, or an unaligned
    transcription of the entire text,
    or a pointer to an unaligned audio file
  </item>
  <phrases>
    Nested XML content to represent the
    phrasal constituents of the text
 </phrases>
</interlinear-text>
```

This structure is repeated at each of the four levels, using the Yidinj example again:

```
<interlinear-text>
 <item type="title">A Yidinj Story</item>
  <phrase>
   <item type="number">98</item>
   <item type="gls">[When Damari eventually
did turn up at the fighting ground, Guyala
asked him:] ’Where have you [come] from?’
</item>
    <words>
     <word>
      <item type="txt">nundu</item>
       <morphemes>
        <morph>
         <item type="gls">you-SA</item>
        </morph>
       </morphemes>
     </word>
     <word>
      <item type="txt">wandam</item>
       <morphemes>
        <morph>
         <item type="gls">where-ABL</item>
        </morph>
       </morphemes>
     </word>
    </words>
   </phrase>
  </phrases>
</interlinear-text>
```

4 Interlinear Text Styles

Now that we have a model of the structure of interlinear text, we demonstrate its adequacy by showing how it can be used in generating a variety of layouts. (This logic is largely analogous to that used in conventional generative grammar: having analysed surface forms and proposed an underlying representation, it is incumbent upon us to provide the rules and constraints which can be applied to generate the original surface forms from the putative underlying representations.) Here we discuss a variety of presentation issues, before defining and demonstrating the mapping using XSL in the following sections.

4.1 Grouping of Content

Generalising from these examples, it should be possible to view an interlinear text with coarser-grained alignment, e.g. combining phrase-level items to form a single text-level item, or combining morpheme-level items to form a single word-level item. More generally, when a text is enriched by the addition of finer-grained segmentation, it should still be possible to view it with coarser alignment. In terms of the hierarchical model, this amounts to taking material from a lower-level set of nodes, concatenating it together, and storing it in a higher-level node. Just as we can ignore the low-level structures, we must also be able to ignore the high-level structures. For example, given an interlinear text it should be possible to extract its words or morphemes in order to construct a word-list. Therefore, it is a requirement on the rendering process that it can ignore aspects of the structure for the purpose of display and alignment.

4.2 Which Rows to Display

Interlinear texts have widely varying levels of detail, ranging from two rows to a dozen. When a text is enriched by the addition of another row of information, it should still be possible to view it in its original form without this extra row. Therefore, it is a requirement on the rendering process that it can omit specific rows from the display.

4.3 Row Styles

Font variation in form may reflect the preference of the author or the requirements of the publisher. It should be possible to view the rows of a given text in different styles without having to manually change the style of every item in the text. Therefore, it is a requirement of the rendering process that it can distinguish different types of rows and display them in user-specified fonts, typefaces, point sizes and so forth.

4.4 Ordering of Rows

There are some widespread conventions concerning the vertical arrangement of the rows of interlinear text. For example, morphemes are usually written underneath the corresponding words , and glosses are usually written underneath the corresponding morphemes. However, there are occasional exceptions to such patterns. At the text level we can observe considerable variation. In some cases, the notes on the text are placed after the phrases of the text, in other places the order is reversed. It should be possible to view a the rows of a given text in different orders without having to manually reorganise the layout of every item. Therefore, it is a requirement on the rendering process that it can distinguish different types of rows and display them in a user-defined sequence.

5 Rendering Interlinear Text with XSL

Our implementation of rendering is based on the Extensible Stylesheet Language (XSL) (Bradley, 2000), which can be used to transform XML documents into other formats. By choosing different stylesheets, or selecting different parameters for a given stylesheet, it should be possible to generate a variety of useful formats, whether for human consumption using a particular technology (e.g. conversion to HTML for delivery to a web browser), or for machine consumption (e.g. conversion to or serialisation of another XML format for delivery to another program).

The transformation performed by this model must accomplish two things: (i) convert the abstract XML representation into a format which specifies grouping, row ordering and styles, and (ii) convert the XML markup into the formatting instructions of some other language like PDF or HTML. The second of these can be further broken down into two stages: conversion to XSL Formatting Objects, and conversion to the delivery format. The full model is shown in Figure 2 below.

On the left we see the abstract representation, which is mapped using one of our stylesheets XSL_1 to a surface representation that fixes the grouping of items, ordering of rows, and so forth. A different choice of stylesheet will result in different groupings and orderings in the surface representation $\mathrm{XML}_{\mathrm{SR}}$. This format may be further transformed using third-party XSL stylesheets ($\mathrm{XSL}_{\mathrm{PUB}}$) to meet the requirements of publishers. We supply another transform $\mathrm{XSL}_{\mathrm{FO}}$ which converts the surface representation into a low-level representation using XSL formatting objects. Third party software can then be used to generate the format to be delivered to the enduser.

The key rendering challenge posed by interlinear text is line-wrapping. A line of interlinear text contains multiple rows, and these must be wrapped as a group, keeping words and their morphological analysis together on the same line. Thus, these multi-row constituents must be treated as indivisible entities. Formatting languages such as HTML (Raggett, Le Hors and Jacobs, 1999), DocBook (OASIS, 2003) and DSSSL (ISO/IEC, 1996) model a document as a collection of blocks (paragraphs, quotes, tables, lists) each containing lines of text. Critically, blocks must appear on a line of their own, or equivalently, lines cannot contain multiple blocks. In order to handle the line wrapping requirements of interlinear text, we need a language which permits inline blocks, such as TeX or XSL-FO (XSL Formatting Objects (Adler et al, 2003)). To facilitate flexible integration with web environment we have chosen to use XSL-FO.

Unfortunately, some XSL-FO engines do not handle inline blocks correctly at the present time (e.g. Apache FOP (The Apache Project, 2003) and XEP (RenderX, 2003)). After experimentation, we have found that XSL Formatter (AntennaHouse, 2003) correctly handles inline blocks, and so we have used it to generate the examples included below. For a discussion of XSL-FO engines and their various conformance levels, see Kimber (2002).

XSL processing of the abstract XML format to specify grouping, row-ordering and styles is quite straightforward. We give three examples to illustrate. The following template matches a phrase, and orders its constituent words before any content at the phrase level (such as a free translation).

```
<xsl:template match="phrase">
 <phrase>
  <xsl:apply-templates select="words"/>
  <xsl:apply-templates select="item"/>
 </phrase>
</xsl:template>
```

The following template matches an interlinear-text, ordering any content of type “title” before the constituent phrases, and ordering any other content (e.g. notes) afterwards.

```
<xsl:template match="interlinear-text">
 <interlinear-text>
  <xsl:for-each
   select="item[@type='title']">
  <title>
   <xsl:value-of select = "." />
  </title>
  </xsl:for-each>
   <phrases>
    <xsl:apply-templates
        select="phrases"/>
   </phrases>
    <xsl:for-each
      select="item[@type!='title']">
      <item>
        <xsl:value-of select = "." />
      </item>
    </xsl:for-each>
  </interlinear-text>
</xsl:template>
```

The following template matches a document and constructs a single interlinear text consisting of just the words (including any morphemes and glosses), and sorts them.

```
<xsl:template match="document">
 <document>
  <interlinear-text>
   <phrases>
    <xsl:for-each
     select="interlinear-text/phrases/
phrase/words/word">
    <xsl:sort select="."/>
     <phrase>
      <words>
       <xsl:copy-of select="."/>
      </words>
     </phrase>
    </xsl:for-each>
   </phrases>
  </interlinear-text>
 </document>
</xsl:template>
```

Using such templates we can generate a variety of document types for external processing (e.g. use by third-party software) or for delivery to end-users.

Further work on the XSL mapping is needed: to support common layout styles and to include additional rendering parameters; to permit display parameters to be stored within the abstract XML representation itself (or in a special XML format declaration file); and to model affixation in order that hyphens are introduced appropriately at morpheme boundaries.

6 Prototype

In demonstrating our model we have adopted a three level architecture which consists of an underlying data representation, a surface display format, and a variant display format. Essentially the processes involved in demonstrating the flexibility of this architecture are the conversion of the underlying data to a surface display, and then converting the surface display to a variant display.

6.1 Underlying Data

The underlying data is interlinear text structured according to our model and expressed in XML. This underlying data can be validated against an existing DTD or schema.

6.2 Surface Display

The surface display is a basic display format corresponding to traditional interlinear text which is enabled by the application of an XSL stylesheet to the underlying data. There are two types of surface display we have identified. The first is a simple type, namely direct application of a single XSL stylesheet to an underlying XML document. The second is more complex, including the parameterisation of user selected display input which in turn affects the XSL stylesheet and the corresponding display of the underlying XML document. These distinctions form the basis of the categorisation of functions.

6.3 Variant Display

The variant display is a customised display format which demonstrates the flexibility of manipulating the underlying data for different display purposes. For this demonstration we have identified a number of desirable variants based on common linguistic data structures.

6.3.1 Simple Display Types

We have identified two simple display types: (i) free translation as separate block and (ii) frame interface based expansion of free translation. In the first variant, we manipulate the surface display to format the free translation as a separate block of text from the interlinear content. In the second, we manipulate the surface display to provide the free translation in a separate frame from the interlinearization, and allow synchronised scrolling and linking between the segments of the free translation and the relevant interlinear segments.

6.3.2 Complex Display Types

We have identified a number of complex display types based on parameterised input. These are tree view or metastructural view; row reordering; optional row display; wordlist linkage; and concordance linkage. The *tree-view* or *metastructural display* essentially allows navigation of the interlinear text using a tree view display format. Individual branches of the tree can be expanded or compressed. This may be useful for structural analysis of the text. The *row reordering display* allows the selection of a preference for the order of the lines of interlinear text, eg display source text first, display source text last. This may be useful in evaluation contexts for back-glossing. The *optional row display* allows a selection of preference for how many interlinear lines are displayed. This may be useful for context where features of interest are identified in the interlinear text (eg syntactic vs morphological vs phonological annotation). The *word list linkage display* allows the selection of a particular word, and the corresponding display of interlinear content for that word. This may be useful for contexts where particular words are of intermittent interest and detected whilst browsing the source or translation text. The *concordance linkage display* allows the selection of a particular, and the corresponding display a list of of all other occurrences of the particular word within the text, including the surrounding words. Any context can be selected, and the complete interlinearization displayed for that context.

6.4 Implementation

We have implemented a prototype which supports the rendering of user-nominated display types. Our implementation is web based, with all user interaction occurring within the browser but reliant on embedded XML rendering capabilities and plugins to handle external application types such as PDF. Our implementation can be described as consisting of three modules, namely the user interface, the parameterisation logic, and the rendering engine.

The user interface is very simple, allowing the user to select the input text from a series of XML interlinear sources, then to select various display types, and finally to select an output format. Underlying each of these choices are a series of parameters, which are processed by a script to determine the display type and result type. These parameters then are passed to the rendering engine, which in combines the interlinear source and the option parameters to generate the appropriate output type which is then sent back to the browser for either direct display or display through a browser plugin.

At a technical level, the user interface itself is written in HTML, while the parameterisation logic is coded in PHP. We have experimented with two server side implementations - one is a wrapper based approach to a Java application executed in the shell; the other is using a Java servlet supported by Apache and Tomcat. (These alternative implementations reflect the lack of a fully-featured rendering engine which services both browser based rendering and plugin-based rendering.

There are a number of extensions to the prototype we hope to implement in the future, including a remotely instantiable rendering engine, the ability for users to contribute their own interlinear texts for rendering, the ability for users to control aspects of the display types, and a wider variety of output formats (eg. JPEG, SVG, or tree diagrams).

7 Future Research

Although we have developed a specification and prototype implementation for interlinear text, there are a number of areas which warrant further research. Here we identify and briefly discuss three of these, namely: architectural extensions; the development of an API for interlinear text manipulation; and the prospects for embedding of the interlinear text model with existing (and ultimately, new) tools. The creation of such tools which are designed for use by the linguist for creating, editing and publishing of interlinear texts represents a significant undertaking, but is based on the foundational work of analysing interlinear text structure, and expressing this in an open format.

7.1 Architectural Extensions

Having discussed the presentational flexibility that the XML-based specification provides, we can now turn to the corresponding architectural flexibility inherent in XML itself. We will consider linkages of interlinear text to higher level linguistic ontologies; as the subject for mining and retrieval in large corpora, such as the Web; and compatibility with other schemas for the representation of materials in an interlinear fashion.

7.1.1 Linguistic Ontologies

Interlinear text typically includes annotations regarding a wide variety of linguistic phenomena. Each annotator typically uses a different labelling inventory for such analysis, and thus automated cross-linguistic enquiry is made significantly more difficult through the disparate labelling of linguistic features. In order to leverage the large amount of annotated interlinear text which exists, there is a requirement for annotation to subscribe to a common ontology of linguistic concepts. Such a linguistic ontology (GOLD) has been defined in Farrar, Lewis and Langendoen (2002) and further refined in Farrar and Langendoen (2003). Our model provides ease of integration of such ontology-based annotation structures, and may provide a leverage point for theory-neutral cross-linguistic enquiry. Further work is required to fully exploit the power of a formally structured interlinear text with embedded ontological annotations.

7.1.2 Text Mining and Retrieval

Although we provide a model for encoding interlinear text and its subsequent manipulation, it is obvious that a vast amount of interlinear material already exists, some of which is in electronic form. In particular, interlinear text is published online through a variety of document types including lexicons, pedagogical materials, language recordings and transcriptions, language descriptions, grammars and scholarly papers about language. For the most part, this interlinear material is locked up in proprietary formats or is expressed only in a loosely structured fashion. Re-use of encoded interlinear text which exists in this form is desirable, but obstructed through formats which do not lend themselves to easy linkage. Some researchers, notably Lewis (2003), have embarked on efforts to identify interlinear text in these contexts, and to retrieve likely examples of interlinear text for re-use. In this particular context, the existence of a model into which existing interlinear forms can be easily translated is a desirable entity. Providing tools exist to manipulate interlinear text compliant with this new model may provide an incentive for linguistic data managers to explore conversion techniques for common proprietary or loosely structured formats. Assuming such conversion is carried out, the standardised body of interlinear text that results can in turn be queried directly for a variety of purposes.

7.1.3 Compatibility with Other Schemata

In our earlier work, we clearly delineated interlinear text from other materials which were expressed in an interlinear format. Models such as those proposed by Brugman (2003) use interlinear text as a representational medium for non-textual data, in particular for multimodal sources such as audio and video. The annotation of such data sources may require variation in the granularity of an interlinear model in order to capture the extra-linguistic dimension of the expression (eg gesture). Whilst we position our model as an interchange (and possibly archival format), we acknowledge that other annotation types require different structures in which to encode annotations. Already we have commenced investigations into the use of XML namespaces to allow both types of annotation and analysis to coexist within a single XML document. Software which is XML namespace-aware can then interpret rows of analysis according to the appropriate textual or multimodal models. Further work in this area is required to ensure tight integration and the development of appropriate conversion tools.

7.2 An API for Interlinear Text Manipulation

We have presented a data model but apart from our discussion of rendering we have not attempted to define the legal operations that construct, access and otherwise manipulate the structures. In future work we plan to define the data types and legal operations formally, independently of their possible XML and XSL representations. This will serve as the basis for object-oriented implementation, for the definition of application programming interfaces, and for research on suitable query languages. More generally, we need to consider interfaces to other data sources such as texts and lexicons.

7.3 Embedding Interlinear Functionality in Application Instances

The model presented here is a specialisation of the interlinear text model presented by Maeda and Bird (2000), and can be represented using annotation graphs. In future work we will implement a tool that is part of the Annotation Graph Toolkit (AGTK, http://agtk.sf.net), building on the existing InterTrans tool (Bird et al, 2002). This will require extending the XML format to support the specification of media file offsets which is likely to be trivial. In addition, translation tools which support conversion between native AGTK and our proposed model format for interchange will facilitate the import of existing data into this new tool.

8 Conclusion

Interlinear text is a highly pervasive data type in the linguistic domain. Although a number of tools have gained widespread acceptance for creating and editing interlinear text, the lack of an open and extensible model has resulted in annotated textual data in interlinear form being tied to particular implementations either as structural or presentation formats. In this paper we have sought to present open, extensible encoding and presentation mechanisms which allow the re-use of interlinear text in a variety of output formats. A significant advantage of adopting an XML-based structural encoding is that interlinear text can potentially be manipulated and queried systematically by any number of tools which subscribe to open standards, whether they have a linguistic lineage or otherwise.

References

Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Tony Graham, Paul Grosso, Eduardo Gutentag, Alex Milowski, Scott Parnell, Jeremy Ridner and Steve Zilles. 2001. *Extensible Stylesheet Language Version 1.0* The World Wide Web Consortium. http://www.w3.org/TR/2001/REC-xsl-20011015/

Antenna House Incorporated. 2003. *XSLFormatter* Antenna House Incorporated. http://www.antennahouse.com/xslformatter.htm

The Apache Project. 2003. *Apache Formatting Objects Processor* The Apache Project. http://xml.apache.org/fop

Steven Bird, Kazuaki Maeda, Xiaoyi Ma, Haejoong Lee, Beth Randall, and Salim Zayat. 2002. *TableTrans, MultiTrans, InterTrans and TreeTrans: Diverse Tools Built on the Annotation Graph Toolkit*. Proceedings of the Third International Conference on Language Resources and Evaluation, Paris: European Language Resources Association, pp 364-370.

Catherine Bow, Baden Hughes and Steven Bird. 2003. *Towards a General Model of Interlinear Text.* Proceedings of the EMELD Language Digitization Project Conference 2003: Workshop on Digitizing and Annotating Texts and Field Recordings. LSA Institute, University of Michigan, Lansing MI USA. http://www.emeld.org/workshop/2003/papers03.html

Catherine Bow, Baden Hughes and Steven Bird. 2003. *The EMELD Model for Interlinear Text* Manuscript in preparation.

Neil Bradley. 2000. *The XSL Companion*. Addison-Wesley.

Hennie Brugman. 2003. *Annotated Recordings and Texts in the DOBES Project* Proceedings of the EMELD Language Digitization Project Conference 2003: Workshop on Digitizing and Annotating Texts and Field Recordings. LSA Institute, University of Michigan, Lansing MI USA. http://www.emeld.org/workshop/2003/papers03.html

DSSSL 1996. *Document Style Semantics and Specification Language* ISO/IEC 10179:1996

The EMELD Project 2000. *Electronic Metastructures for Endangered Language Documentation* http://www.emeld.org

Scott Farrar and Terry Langendoen. An Ontology for Linguistic Annotation. To appear in *GLOT International*.

Scott Farrar, William Lewis and Terry Langendoen 2002. *A Common Ontology for Linguistic Concepts.* Proceedings of the Knowledge Technologies Conference, March 10-13 2002, Seattle.

Baden Hughes, Steven Bird and Catherine Bow. 2003. *Interlinear Text XML and XSL Demonstration Handout.* Proceedings of the EMELD Language Digitization Project Conference 2003: Workshop on Digitizing and Annotating Texts and Field Recordings. LSA Institute, University of Michigan, Lansing MI USA. http://www.emeld.org/workshop/2003/papers03.html

W. Eliot Kimber 2002. *Production Quality XSL-FO*. http://xml.coverpages.org/KimberProductionQuality-XSL-FO.html

William Lewis. 2003. *Migrating and Mining Interlinear Text.* Proceedings of the EMELD Language Digitization Project Conference 2003: Workshop on Digitizing and Annotating Texts and Field Recordings. LSA Institute, University of Michigan, Lansing MI USA. http://www.emeld.org/workshop/2003/papers03.html

Kazuaki Maeda and Steven Bird. 2000. *A Formal Framework for Interlinear Text*. Proceedings of the Workshop on Web-based Documentation and Description, Philadelphia, USA; December 12-15, 2000.

Organisation for the Advancement of Structured Information Standards (OASIS). 2003. *DocBook* http://www.oasis-open.org/docbook

Dave Raggett, Arnaud Le Hors and Ian Jacobs 1999. *The Hypertext Markup Language Version 4.1* The World Wide Web Consortium. http://www..w3.org/TR/html4

RenderX. 2003. *XEP XSL Rendering Engine* RenderX Inc. http://xep.xattic.com/

9 Acknowledgements

This research in this paper has been supported by the National Science Foundation under Grant No. 0094934 (Electronic Metastructure for Endangered Languages Data).

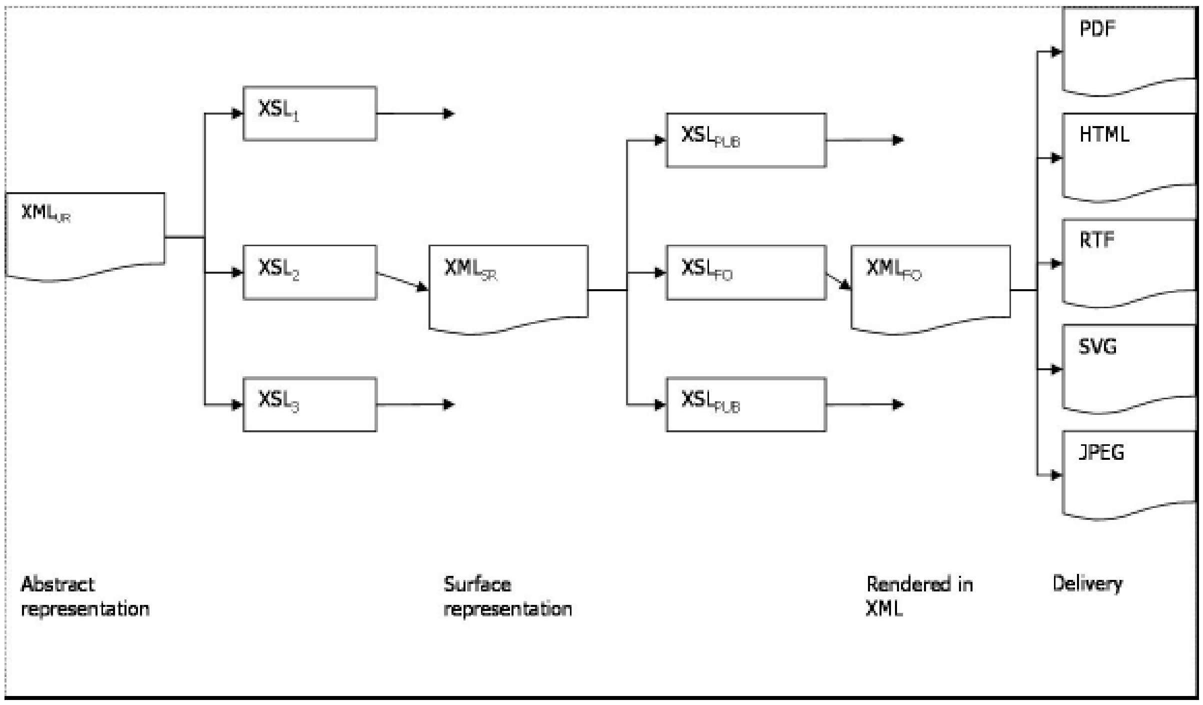

Figure 2: The Rendering Model