Journal of the Text Encoding Initiative

Issue 6 | December 2013
Selected Papers from the 2012 TEI Conference

Resolving the Durand Conundrum

Lou Burnard

Electronic version
URL: http://journals.openedition.org/jtei/842
DOI: 10.4000/jtei.842
ISSN: 2162-5603

Publisher
TEI Consortium

Electronic reference
Lou Burnard, « Resolving the Durand Conundrum », *Journal of the Text Encoding Initiative* [Online], Issue 6 | December 2013, Online since 04 September 2017, connection on 19 April 2019. URL : http://journals.openedition.org/jtei/842 ; DOI : 10.4000/jtei.842

Resolving the Durand Conundrum

Lou Burnard

1. Resolving the Durand Conundrum

1 The Durand conundrum is a jocular name for a serious question first raised by David Durand when the current TEI ODD XML format was being finalized at a meeting of the TEI Technical Council held in Gent in May 2004. In the original TEI ODD system, used for the production of all versions before P4, content models and system entities were declared using SGML syntax, which was then wrapped in ODD-defined containers of various kinds. When the TEI moved to XML, the line of least resistance was to re-express those same SGML rules using RELAX NG, thus perpetuating in ODD XML a hybrid beast of a language, in which everything except element content models was expressed in our own XML vocabulary, while element content models used the RELAX NG XML vocabulary in the RELAX NG namespace. David pointed out, quite reasonably, that this was a lazy compromise of a solution: like other XML vocabularies, the TEI was perfectly hospitable to other namespaces, so we could equally well embed our TEI additions within a natively RELAX NG document. A similar suggestion is made in Eric Van der Vlist's *RELAX NG*[1] (2004) which proposes a hybrid language called 'Exemplotron' in which the documentation is expressed using the XHTML vocabulary, the document grammar is expressed using RELAX NG, and additional constraints are expressed using Schematron.[2] What added value do we derive from the use of the ODD system?

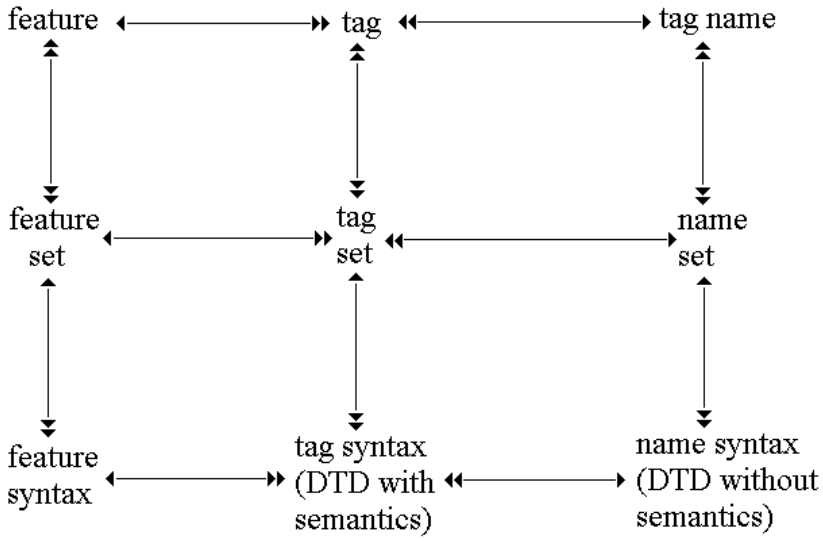
2 At the time, we called this a conundrum because we couldn't come up with any entirely convincing answer to the question, beyond a vague feeling that a hybrid system is just not good engineering, and a desire to retain the additional layer of abstraction provided by the TEI's independence of any single schema language. The ODD language expresses a conceptual model for digitally stored text, and we would prefer to manipulate such texts entirely in terms of that model, as far as possible, without having to think too much about other formal languages into which that model can be mapped. Even with the current system, the need to support different interpretations of class references or interleaving of

child elements poses problems derived from the fact that such operations are not fully expressible using the current ODD language. But the Durand conundrum can be resolved in two ways: one would indeed be to re-express everything currently expressed using ODD in a combination of non-TEI XML vocabularies, embedding TEI and Schematron elements within another XML language such as RELAX NG; the other would be to expand ODD to enable it to define content models natively, without having to recourse to RELAX NG. This article explores the second possibility.

2. Some Background

3 The TEI began as a conscious attempt to *model* existing and future markup systems. The original TEI editors, Michael Sperberg-McQueen and I, had spent much of our careers to date trying to find satisfactory ways of expressing the rugosities of typical humanities datasets using the database modeling techniques common in the IT industry at that time. We naturally turned to them to help us draw up a formal model of textual features and their representations in different markup schemes. The following figure, taken from an early paper we wrote on the topic (1989), typifies our approach: it distinguishes sharply between the features perceived in a text, their representation by the application of tags, and the names that might be used for those tags.

Figure 1. Abstract model from TEI EDW05, 1989.



4 This exercise in modeling started to become more than theoretical quite early in the life of the TEI, notably during 1991, when the TEI's initial workgroups started to send in their proposals for textual features which they felt had to be distinguished in any sensible encoding project. It rapidly became apparent that something better than a hypercard stack or relational database would be needed to keep track of the tags they were busy inventing and the meanings associated with them. In particular, something able to *combine* text and formal specifications in a single SGML document was needed.

Fortunately Donald Knuth had been here before us, with his concept of "literate programming".[3]

In the autumn of 1991, we started seriously thinking about ways of implementing the idea of a single DTD which could support both the documentation of an encoding scheme and its expression as a formal language. Our thoughts were necessarily constrained to some extent by the SGML technology at our disposal, but we made a considered effort to abstract away from that in the true spirit of literate programming as Knuth eloquently defines it: "Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do" (Knuth 1984, 97). The documentation for each element in the proposed system thus needed to provide informal English-language expressions about its intended function, its name and why it was so called, the other elements it was associated with in the SGML structure, usage examples, and cross-references to places where it was discussed, along with formal SGML declarations for it and its attribute list. Relevant portions of these "tag documents" could then be extracted into the running text, and the whole could be reprocessed to provide reference documentation as well as to generate document type declarations for the use of an SGML parser. The following figure shows a typical example of such a tag document.

Figure 2. Tagdoc for `<resp>` element in P2.

```
<tagdoc usage=rwa id="resp"><gi>resp</gi>
<name>statement of responsibility</name>
<desc>supplies information about someone other than an author,
sponsor, funder or principal researcher responsible for the
intellectual content of a text, edition, recording, or
series.</desc>
<attlist></attlist>
<exemplum><eg><![ CDATA [
  <resp><role>transcribed from original ms</role>
        <name>Claus Huitfeldt</name>
  </resp>
]]>
</eg></exemplum>
<exemplum><eg><![ CDATA [
  <resp><role>converted to SGML encoding</role>
        <name>Alan Morrison</name>
  </resp>
]]>
</eg></exemplum>
<remarks></remarks>
<part>auxiliary tag set for TEI headers</part>
<classes>
<files names='teihdr2'>
<datadesc></datadesc>
<parents>editionStmt recording seriesStmt titleStmt</parents>
<children>role name</children>
<elemdecl>
<![ CDATA [
<!ELEMENT resp          - o  ((role & name)+)                  >
]]>
</elemdecl>
<attldecl>
<![ CDATA [
<!ATTLIST resp              %a.global;                         >
]]>
</attldecl>
<xref target='hd21'>
</tagdoc>
```

6 Note here how the SGML declarations are embedded as floating CDATA marked sections, effectively isolating them from the rest of the document, and thus making it impossible to process them in any way other than by simple inclusion. Such refinements as, for example, checking that every element referenced in a content model has its own specification are hard or impossible. There is also ample scope for error when the structural relationships amongst elements are redundantly expressed both in DTD syntax, and in human readable form using the `<parents>` and `<children>` elements. Nevertheless, this system[4] served the TEI well, both in its original form and in later developments of it.

7 At P5 (2007), the TEI switched to using RELAX NG as the primary means of declaring content models, both within the element specifications which had replaced the old tag documents as input, and as output from the schema generation process. As a separate processing step, XML DTDs are also generated from this same source, while W3C schema is generated from the RELAX NG outputs using James Clark's trang processor. Another major change at TEI P5 was the introduction and extensive use of model classes as a means of implementing greater flexibility than had been achievable by using SGML parameter entities. Both of these changes (along with those consequent on an extensive internationalization effort) are reflected in the partial TEI P5 specification for the `<respStmt>` element shown in the following figure:

Figure 3. Parts of `<respStmt>` element in P5 (XML).

```
<desc versionDate="2007-01-21" xml:lang="it">fornisce una dichiarazione di r
  responsabile del contenuto intelletuale di un testo, curatela, registrazio
  in cui gli elementi specifici per autore, curatore ecc. non sono sufficien
<classes>
  <memberOf key="att.global"/>
  <memberOf key="model.respLike"/>
  <memberOf key="model.recordingPart"/>
</classes>
<content>
  <choice xmlns="http://relaxng.org/ns/structure/1.0">
    <group>
      <oneOrMore>
        <ref name="resp"/>
      </oneOrMore>
      <oneOrMore>
        <ref name="model.nameLike.agent"/>
      </oneOrMore>
    </group>
    <group>
      <oneOrMore>
        <ref name="model.nameLike.agent"/>
      </oneOrMore>
      <oneOrMore>
        <ref name="resp"/>
      </oneOrMore>
    </group>
  </choice>
</content>
<exemplum xml:lang="en">
  <egXML xmlns="http://www.tei-c.org/ns/Examples">
    <respStmt>
      <resp>transcribed from original ms</resp>
      <persName>Claus Huitfeldt</persName>
    </respStmt>
  </egXML>
</exemplum>
<exemplum versionDate="2008-04-06" xml:lang="fr">
  <egXML xmlns="http://www.tei-c.org/ns/Examples">
    <respStmt>
      <resp>Nouvelle édition originale</resp>
      <persName>Geneviève Hasenohr</persName>
    </respStmt>
  </egXML>
```

8 Where TEI P2 had used embedded DTD language to express content models, TEI P4 had expressed them using string fragments still recognizably derived from SGML DTD language. In TEI P5, we moved to using RELAX NG in its own namespace, thus placing that

schema language in a privileged position, and inviting the question expressed above as the Durand conundrum.

3. What's not ODD?

9 In the current source of TEI P5, there is extensive use of several different XML vocabularies:

- Examples in TEI P5 are presented as if they belonged to some other "TEI Example Namespace";
- Element content models are expressed using a subset of RELAX NG, as discussed in the previous section;
- Datatypes are expressed in a variety of ways, mapping either to built-in W3C datatypes (as defined in the W3C Schema Language) or to RELAX NG constructs;
- Additional semantic constraints (for example, co-dependence of attributes and element content) are expressed using ISO Schematron rules.

10 Everything else in a TEI-conformant ODD specification uses only constructs from the TEI namespace. In this paper, we will argue for a further extension of the ODD language to replace several of the cases listed above.

3.1 Element Content Models

11 ODD was originally intended to support the *intersection* of the content models definable using three different schema languages. In practice, this reduced our modeling requirements quite significantly. Support for DTD schema language in particular imposes many limitations on what would otherwise be possible, while the many additional facilities provided by W3C Schema and RELAX NG for content validation are hardly used at all (though some equivalent facilities are now provided by the `<constraintSpec>` element). A few years ago, the demise of DTDs was confidently expected; in 2013, however, the patient remains in rude health, and it seems likely that support for DTDs will continue to be an ongoing requirement, even though support for P4 was formally withdrawn at the end of 2012. We therefore assume that whatever mechanism we use to specify content models will need to have the following characteristics:

- alternation, repetition, and sequencing of individual elements, element classes, or sub-models (groups of elements) are required;
- only one kind of mixed content model—the classic `(#PCDATA | foo | bar)*`—is permitted;
- the SGML ampersand connector—`(a & b)` as a shortcut for `((a,b) | (b,a))`—is not permitted;
- a parser or validator is not required to do look-ahead and consequently the model must be deterministic, that is, when applying the model to a document instance, there must be only one possible matching label in the model for each point in the document.

12 These requirements can easily be met by the following small incremental changes to the ODD language:

- **Specification**
  - At present, references to content model components use the generic `<rng:ref>` element. As a consequence, naming conventions have been invented to

distinguish, for example, references to an element or attribute class (name starts with 'model.' or 'att.') from references to a predefined macro (name starts with 'macro.') or from references to an element (name starts with something other than 'model.' or 'macro.'). Although these name changes are purely a matter of convenience, we suggest that it would be better to use the existing XML ODD elements `<elementRef>`, `<classRef>`, and `<macroRef>`.
For example, `<rng:ref name="model.pLike"/>` becomes `<classRef key="model.pLike"/>`.

- **Repeatability**
  - In RELAX NG, this is indicated by special-purpose grouping elements `<rng:oneOrMore>` and `<rng:zeroOrMore>`. We propose to replace these by the use of attributes `@minOccurs` and `@maxOccurs`, which are currently defined locally on the `<datatype>` element. Making these also available on `<elementRef>`, `<classRef>`, and `<macroRef>` elements gives more delicate and consistent control over what is possible within the components of a content model. It is most easily achieved by defining a new attribute class (say, `att.repeatable`).
- **Sequence and alternation**
  - Sequencing and alternation are currently indicated by elements defined in the RELAX NG namespace (`<rng:choice>`, `<rng:group>`, etc.) We replace these by similar but more constrained TEI equivalents: `<sequence>`, which operates like `<rng:group>` to indicate that its children form a sequence within a content model, and `<alternate>`, which operates like `<rng:choice>` to supply a number of alternatives.
- **Text nodes**
  - A special purpose `<pcdata>` element could be added to indicate that the content model permits a text node at this point. (The choice of name is dictated partly by the fact that the element `<text>` already exists in the TEI namespace, but with a rather different sense.) An alternative, and simpler, method would be be to follow the W3C Schema approach and define an attribute `@mixed` for each container element. This would have the advantage of making it possible to prevent use of a pcdata node in places where it is not permitted by the rules of XML.

13 We now provide some simple examples, showing how some imaginary content models expressed using XML DTD syntax might be re-expressed with these elements.

14 In this example `((a, (b|c)*, d+), e?)` we have a sequence containing a single element, followed by a repeated alternation, a repeated element, and an optional element. This would be expressed as follows:

```
<content>
  <sequence>
    <sequence>
      <elementRef key="a"/>
      <alternate minOccurs="0" maxOccurs="unbounded">
        <elementRef key="b"/>
        <elementRef key="c"/>
      </alternate>
      <elementRef key="d" maxOccurs="unbounded"/>
    </sequence>
    <elementRef key="e" minOccurs="0"/>
  </sequence>
</content>
```

15 Repetition can be applied at any level. In `((a, (b*|c*))+`, for example, we have a repeated sequence. This would be expressed as follows:

```
<content>
  <sequence maxOccurs="unlimited">
    <elementRef key="a"/>
    <alternate>
      <elementRef key="b" minOccurs="0" maxOccurs="unlimited"/>
      <elementRef key="c" minOccurs="0" maxOccurs="unlimited"/>
    </alternate>
  </sequence>
</content>
```

16 A mixed content model such as `(#PCDATA | a | model.b)*` might be expressed as follows:

```
<content>
  <alternate minOccurs="0" maxOccurs="unlimited" mixed="true">
    <elementRef key="a"/>
    <classRef key="model.b"/>
  </alternate>
</content>
```

17 References to model classes within content models pose a particular problem of underspecification in the current ODD system. In the simple case, a reference to a model class may be understood as meaning any one member of the class, as assumed above. Hence, supposing that the members of class `model.ab` are `<a>` and `<b>`, a content model

```
<content>
  <classRef key="model.ab" maxOccurs="unlimited"/>
</content>
```

is exactly equivalent to

```
<content>
  <alternate maxOccurs="unlimited">
    <elementRef key="a"/>
    <elementRef key="b"/>
  </alternate>
</content>
```

18 However, sometimes we may wish to expand model references in a different way. We may wish to say that a reference to the class `model.ab` is not a reference to any one of its members, but to a sequence of all of its members, or to a sequence in which any of its members may appear, and so forth. This requirement is handled in the current ODD system by over-generating all the possibilities, again using a set of naming conventions to distinguish amongst them. We propose instead to control this behaviour by means of a new `@expand` attribute on `<classRef>` that behaves in much the same way as the existing `@generate` on `<classSpec>`, but with the advantage of being usable at the instance level.

19 For example,

- `<classRef key="model.ab" expand="sequence"/>` is interpreted as `a,b`
- `<classRef key="model.ab" expand="sequenceOptional"/>` is interpreted as `a?,b?`
- `<classRef key="model.ab" expand="sequenceRepeatable"/>` is interpreted as `a+,b+`
- `<classRef key="model.ab" expand="sequenceOptionalRepeatable"/>` is interpreted as `a*,b*`

20 Note that the ability to specify repetition at the individual class level gives a further level of control not currently possible. For example, a model containing no more than two consecutive sequences of all members of the class `model.ab` could be expressed quite straightforwardly:

```
<content>
  <classRef key="model.ab" maxOccurs="2" expand="sequence"/>
</content>
```

3.2 Datatyping and Other Forms of Validation

21 Validation of an element's content model is but one of many different layers of validation that a TEI user may wish to express in their ODD specification. As noted above, the

current system also provides mechanisms to constrain the possible values of attributes rather more tightly than all schema languages permit by means of datatyping and also, increasingly, by explicit constraints expressed using languages such as ISO Schematron. It seems reasonable to ask how many of these additional layers may be incorporated into our proposed new vocabulary. The vast majority of TEI attributes currently define their possible values by reference to a datatype macro defined within the ODD system. These definitions are in turn mapped either to a W3C Schema datatype (as is customary in RELAX NG) or to an expression in RELAX NG syntax. This indirection allows the schema builder to add a small amount of extra semantics to an underlying "bare" datatype. For example `data.duration.iso`, `data.outputMeasurement`, `data.pattern`, `data.point`, `data.version`, and `data.word` all map to the same datatype CDATA in XML DTD, and to various TEI-defined regular expressions in RELAX NG or W3C Schema. As their names suggest, however, each of these TEI datatypes has a subtly different intended application, which an ODD processor may use in deciding how to present the corresponding information independently of the mapping to a formal schema language, although in the current architecture this information is lost once the target formal schema has been generated.

22 Given the existence of this TEI abstraction layer, it seems unnecessary to propose further change to the way attribute values are constrained in the ODD system. At the time of writing, there are still a few attributes whose values are expressed directly in RELAX NG syntax, but that is a corrigible error in the Guidelines source code.

23 The most commonly used datatype macro is `data.enumerated`, which maps to another frequently used datatype `data.name`, and thence to the underlying RELAX NG datatype for an XML Name. The intended difference between an enumeration and a name is, of course, that a (possibly closed) list of possible values should always be provided for the former but is not required for the latter. In the ODD system, for every datatype declared as `data.enumerated`, a sibling `<valList>` element should be provided to enumerate and document all or some of the possible values for this attribute. This ability to constrain and document attribute values is of particular interest because it permits TEI schema-specifiers to define project-specific restrictions and semantics considerably beyond those available to all schema languages.

24 A further layer of constraint specification is provided by the `<constraintSpec>` element which may be used to express any kind of semantic constraint, using any suitable language. The ISO-defined Schematron language is currently used to express such constraints, for example to replace as many as possible of the informally expressed rules for good practice which have always lurked in the Guidelines prose. This facility would allow us to specify, for example, the co-occurrence constraint mentioned in the previous paragraph (that the specification for an attribute with a declared datatype of `data.enumerated` should also contain a `<valList>`). It also allows an ODD to make more explicit rules such as "a `<relatedItem>` element must have either a `@target` attribute or a child element" or "the element indicated by the `@spanTo` attribute must follow the element carrying it in document sequence", which are hard or impossible to express in closed grammar-based schema languages.

25 For our present purposes, it is important to note that the `<constraintSpec>` element was designed to support any available constraints language. Although the current generation of ODD processors assume the use of ISO Schematron, there is no reason why future versions should not switch to using different such languages as they become

available without affecting the rest of the ODD processing workflow or the ODD language itself. As such, we see no need to modify our proposals to take this level of validation into account.

4. Discussion

26 The ideas presented here were first sketched out in the summer of 2012, and greeted positively at the ODD Workshop held later that year following the DH 2012 conference in Hamburg. An earlier version of this paper was presented at the TEI Conference in November 2012. In this section we briefly discuss some of the comments received.

27 At first blush, our proposals seem to flout the TEI philosophy of not re-inventing the wheel. The TEI does not and should not take on itself the task of inventing a new XML vocabulary for such matters as mathematics or musical notation where perfectly acceptable and well established proposals are already in place. However, the TEI has arguably already gone down the road of defining some aspects of its own schema language (for example, by providing constructs for representing element and attribute classes, and for associating attribute lists and value lists with element declarations), and this proposal simply continues along the same path. It should also be noted that there are three competing standards for formal schema language in the marketplace (DTD, RELAX NG, W3C Schema) each with its own advantages. By making ODD independent of all three, we make it easier to profit from the particular benefits of each, as well as providing the ability to document intentions not necessarily expressible using any of them.

28 Resolving the Durand conundrum in this way, rather than taking the alternative approach of embedding TEI documentation elements in the RELAX NG namespace, is clearly a compatible expansion of the current scheme rather than an incompatible change of direction which would break existing systems or documents.

29 Because this proposal makes it possible for ODD to support features of current or future schema languages beyond those provided by the current subset, we suggest that it reasserts one of the founding objectives of the TEI, expressed in the Poughkeepsie Principles as follows: "'Compatibility with existing standards and practice' is to be sought, but (as its rank suggests) not at the expense of the other design goals. The standards most relevant to this goal are SGML and existing applications of SGML, as well as the standards now being developed for page description and similar applications. The Text Encoding Initiative will develop a conforming SGML application, if it can meet the needs of researchers by doing so. Where research needs require constructs unavailable with SGML, however, research must take precedence over the standard" (TEI 1988).

30 As a concrete example, consider the occasionally expressed desire to constrain an element's content to be a sequence of single specified elements appearing in any order, that is, to define a content model such as `(a,b,c,d)` but with the added proviso that the child elements may appear in any order. In SGML, the ampersand operator allowed something like this; in RELAX NG the `<interleave>` element may be used to provide it; but there is no equivalent feature in W3C Schema or XML DTD languages, and we have therefore not proposed it in our list of requirements above.

31 Suppose, however, that the Council decided this facility was of such importance to the TEI community that it should be representable in TEI ODD. It would be easy enough to add a new grouping element such as `<interleave>` (or add an attribute `@preserveOrder`

taking values "true" or "false" to our existing proposed <sequence> element) to represent it. Generating a RELAX NG schema from such an ODD would be simple; for the other two schema languages, one could envisage a range of possible outcomes:

1. an ODD processor might simply reject the construct as infeasible;
2. an ODD processor might over-generate; that is, it will produce code which validates everything that is valid according to the ODD, but also other constructs that are not so valid;
3. an ODD processor might over-generate in that way, but in addition produce schematron code to remove "false positives".

32 For example, consider the following hypothetical ODD.

```
<content>
  <interleave>
    <elementRef key="a"/>
    <elementRef key="b" maxOccurs="2"/>
    <elementRef key="c"/>
  </interleave>
</content>
```

33 In XML DTD or W3C schema languages (which lack the <rng:interleave> feature), an ODD processor can represent these constraints by generating a content model such as (a|b|c)+ and at the same time generating additional Schematron constraints to require the presence of no more than one <a> or <c> and up to two <b>s. An extra twist, in this case, is that if there are more than two <b> elements, they must follow each other.

34 As a second example, consider the need for contextual variation in a content model. For example, a <persName> appearing inside a "data-centric" situation, such as a <listPerson> element, is unlikely to contain elements such as <del> or <corr> that are entirely appropriate (and very useful) when identifying names within a textual transcription. In a linguistic corpus, it is very likely that the child elements permitted for <p> elements within the corpus texts will be quite different from those within the corpus header—the latter are rather unlikely to include any part-of-speech tagging, for example.

35 At present only ISO Schematron rules allow us to define such contextual rules, although something analogous to them is provided by the XSD notion of base types. It is not hard, however, to imagine a further extension to the ODD language, permitting (say) an XPath-valued @context attribute on any <elementRef>, <macroRef>, or <classRef> restricting its applicability. Thus, the content model for <p> might say something like

```
<content>
  <elementRef key="s" context="ancestor::text" maxOccurs="unlimited"
minOccurs="1"/>
  <macroRef key="macro.limitedContent" context="ancestor::teiHeader"/>
</content>
```

to indicate that a <p> within a <text> element must contain one or more <s> elements only, whereas one within a TEI Header must use the existing macro definition limitedContent.

36 Since presentation of this paper, an experimental implementation of its proposals has been effected by Sebastian Rahtz at Oxford University. Automated conversion of all existing TEI element specifications to what we are now calling "Pure ODD" has been successfully carried out, and the resulting sources have been used to generate RELAX NG, XML DTD, and W3C Schema source. The schemas generated also pass all previous tests using our existing processing tools.

37 The fact that these three closed-schema languages vary so much in their feature sets suggests that it is indeed prudent to define TEI content models in a way that is independent of all of them. So far we have been cautious in providing only an intersection of their features; the existence of Pure ODD makes possible a more adventurous synthesis in which we select those features most appropriate to the needs of the TEI user community. It also provides the groundwork for a completely new approach when or if a new formalism replaces XML as the language of choice for representing structured textual data. Mapping our TEI abstract language to other formalisms will be an interesting challenge.

BIBLIOGRAPHY

Knuth, Donald E. 1984. "Literate Programming." *The Computer Journal* 27(2): 97-111. doi: 10.1093/comjnl/27.2.97.

Sperberg-McQueen, C.M., and Lou Burnard. November 1989. rev September 1991. "Notes on Features and Tags : TEI EDW5."

TEI (Text Encoding Initiative). 1988, last revised January 9, 1990. "Design Principles for Text Encoding Guidelines (TEI ED P1)". http://www.tei-c.org/Vault/ED/edp01.htm#b2b1b3b3b7b6b32.

van der Vlist, Eric. 2004. *RELAX NG*. Sebastopol, CA: O'Reilly Media.

NOTES

1. http://books.xmlschemata.org/relaxng/page2.html.

2. See further http://examplotron.org.

3. "Literate programming is a methodology that combines a programming language with a documentation language, thereby making programs more robust, more portable, more easily maintained, and arguably more fun to write than programs that are written only in a high-level language." (http://www-cs-faculty.stanford.edu/~uno/lp.html)

4. The first specification for a full ODD system is to be found in TEI Working Paper ED W29, available from the TEI archive at http://www.tei-c.org/Vault/ED/edw29.tar. It defines a set of extensions to the existing `tiny.dtd` (an early version of a simple TEI-compliant authoring schema, not unlike TEI Lite), which adds new elements for documenting SGML fragments, elements, and entities. It also specifies the processing model which the markup was intended to support. An ODD processor was required to extract SGML DTD fragments; generate reference documention (REF) form; and generate running prose (P2X). A processor to carry out the reverse operation (that is, generate template ODD specifications from existing DTD fragments) is also described. Although intended for the use of TEI Workgroups, in practice ODD processors built according to this model were used only by the TEI editors.

ABSTRACTS

This paper proposes a minor but significant modification to the TEI ODD language and explores some of its implications. Can we improve on the present compromise whereby TEI content models are expressed in RELAX NG? A very small set of additional elements would permit the ODD language to cut its ties with any existing schema language, and thus permit it to support exactly and only the subset or intersection of their facilities which makes sense in the TEI context. It would make the ODD language an integrated and independent whole rather than an uneasy hybrid, and pave the way for future developments in the management of structured text beyond the XML paradigm.

INDEX

Keywords: text encoding, ODD, XML, schema design