Reviewing the TEI ODD system

Sebastian Rahtz

IT Services, University of Oxford
sebastian.rahtz@it.ox.ac.uk

Lou Burnard

Lou Burnard Consulting
lou.burnard@retired.ox.ac.uk

ABSTRACT

For many years the Text Encoding Initiative (TEI) has maintained a specialised high-level XML vocabulary in the 'literate programming' paradigm to define its influential *Guidelines*, from which schemas or DTDs in other schema languages are derived. This paper reviews the development of this vocabulary, known as ODD (for 'One Document Does it all'). We discuss some problems with the language, and propose solutions to make it more complete and extensible.

Categories and Subject Descriptors

H.3.7 [**Information Systems**]: Information Storage and Retrieval, Digital Libraries

General Terms

Standardization

Keywords

TEI ODD

1. INTRODUCTION

The Text Encoding Initiative (TEI) began in the late 1980s as a project to provide a consistent and extensible encoding system for digital text of all kinds. By 1991, when the TEI's initial workgroups started to send in their proposals for textual features which they felt really had to be distinguished in any sensible encoding project, it had become clear that something more than a database would be needed to keep track of the tags they were busy inventing, and the meanings associated with them. At that time, it was envisaged that the recently standardized SGML markup system ([3] and [2]) would provide the metalanguage in which those proposals would be expressed (though it is worth noting that even at this stage the TEI considered SGML to be only one possible means for such expression). However, there was no system readily available which combined formal specifications and descriptive explanatory prose in a single document in the spirit of Donald Knuth's concept of 'literate programming'.[1]

DocEng'13, September 10–13, 2013, Florence, Italy.

ACM 978-1-4503-1770-2/13/09 ...$15.00.
http://dx.doi.org/10.1145/2494266.2494321.

In 1991, the TEI editors (Michael Sperberg-McQueen and Lou Burnard) developed the idea of using a single DTD to support both the documentation of an encoding scheme and its expression as a formal language. Like that of other contemporary SGML documentation systems such as Majour, they anticipated a need for documentation about each element in the system, providing English language expressions about its intended function, its name, why it was so called, the other elements it was associated with, usage examples, and cross-references to places where it was discussed. This would however be combined with formal SGML declarations both for the element and for 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. SGML declarations were embedded as 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. This system was named ODD ('One Document Does it all'), and versions of it were used in the definition and production of the first four releases of the TEI Guidelines (1992, 1994, 1996, 2000).

At the fifth major revision however (P5, released in 2007 after several further years of development; [7]), the TEI switched to XML, and to using RELAX NG[4] as the primary means of declaring the content model for each element specification. Pure RELAX NG schemas and XML DTDs were directly generated from this source, while W3C schemas were generated from the RELAX NG output in a second pass, using James Clark's trang processor. Another major change at TEI P5 was the extensive use of model classes as a more expressive and more flexible mechanism for generalization than had been possible using SGML parameter entities.

If there remained a problem with the ODD language, it was the continued presence of another formal language (now RELAX NG rather than SGML) within it. It was noted by the overseeing committee on the TEI Consortium (its Technical Council), notably in a critique by David Durand, that this was a compromise solution: the system neither used the full power of RELAX NG (eg to offer an attribute or child element as alternatives), nor attempted to make the TEI independent of current schema languages. Clearly, there were two ways of addressing this situation. One would be to use pure RELAX NG for *all* parts of the schema definition, embedding TEI documentations and examples in their own namespace within a native RELAX NG document.[1] Alternatively, the system should deploy native TEI constructs with an equivalent expressivity to the subset of RELAX NG currently in use. We present below an im-

[1]This is the approach taken for example in Eric Van der Vlist's hybrid 'Exemplotron'; see further `http://examplotron.org`

plementation of the second approach, by which content models are described in pure TEI.

2. THE CURRENT ODD LANGUAGE

The TEI *Guidelines* define a language for describing, in a single XML document, everything needed to define and describe a schema, from which multiple outputs may be generated. This language (first described in [5], and subsequently refined somewhat for the final release of TEI P5) is not tied to the TEI and can be used to define schemas for any XML language.[2] The outputs may include:

1. formal reference documentation for elements, attributes, element classes, patterns, etc.
2. detailed descriptive documentation, embedding some parts of the formal reference documentation, such as summary lists of tags
3. declarative code for one or more XML schema languages, specifically RELAX NG or W3C Schema.
4. declarative code for fragments which can be assembled to make up an XML Document Type Declaration.
5. ad hoc generated code, such as lists of elements which cannot contain character data, for use by XML processing tools.

The input required to generate these outputs consists of running prose, and special-purpose elements documenting the components (elements, classes, etc.) which are to be declared in the chosen schema language. All of this input is encoded in TEI XML. The same language can also be used to describe a *subset* or customization of the TEI. To make life slightly easier for users, the 560 elements of the TEI are grouped into 22 high-level modules; a typical customization will specify the modules it needs by providing several `<moduleRef>` elements within a top-level `<schemaSpec>` element, whose @*start* attribute specifies the elements allowed as the root of a document.

```
<schemaSpec start="TEI" ident="test1">
 <moduleRef key="core"/>
 <moduleRef key="header"/>
 <moduleRef key="figures"/>
 <moduleRef key="textstructure" include="body div"/>
</schemaSpec>
```

In the example above, the optional @*include* attribute says precisely which elements from a module are needed (where the default is all members). We may also, however, make more granular changes:

```
<schemaSpec start="TEI" ident="test1">
 <moduleRef key="core"/>
 <moduleRef key="header"/>
 <moduleRef key="textstructure" include="body div"/>
 <elementSpec ident="dimensions" mode="change">
  <attList>
   <attDef ident="type" mode="delete"/>
  </attList>
 </elementSpec>
</schemaSpec>
```

In this case the `<dimensions>` element is changed (using the @*mode* attribute), by having an `<attDef>` deleted, again using @*mode* — this can take the values 'change', 'delete', 'add' and 'replace' at most points in the specification, giving a high degree of control. Of course, this also makes it possible to create a schema which conflicts with the TEI. For that reason, TEI P5 included for the first time a detailed discussion of the notion of TEI conformance, which declares some constraints on the type of modifications which may be made. One important principle is that a TEI-conformant modification may not alter the semantics of existing TEI elements; another is that only schemas which are *more specific* than the original can be regarded as TEI-conformant.

Other facilities in the language allow the user to rename elements (eg to internationalize them, while keeping the link to the original name), to encode equivalences with other languages (eg mapping the meaning of TEI elements to ontologies like CIDOC CRM), and to define named groups of customization.

The success of the ODD language is that it enables developers to create rich and complex schemas with a minimum of notation, while encouraging and facilitating the production of high-quality documentation and examples which is also expressive of the specific needs of a project.

3. THE ODD TOOLS

An ODD processor needs to be able to assemble all the components referenced or directly provided, resolve multiple declarations, emit a schema in one or more formal languages, and create selected documentary components. The majority of this work may be accomplished by means of XSLT transforms. The TEI Consortium maintains an extensive open source library of XSLT stylesheets to manage all the necessary tasks.[3] The same stylesheet library underpins a number of transformation tools, for example those made available within the oXygen editor, and is also at the heart of OxGarage.[4]

For non-expert users, writing specifications in XML is not a simple task. Recognizing this, the TEI also provides a graphical interface to the creation of TEI customizations in a web-based editor.[5]

4. COMPLETING THE LANGUAGE

In the current ODD language, the detailed content model of an element is expressed in RELAX NG embedded inside TEI markup. Can we instead define a set of additional elements which would permit the ODD language to cut its ties with existing schema languages, and make it an integrated and independent whole? We describe below the additional features needed to achieve this goal.

ODD currently supports the *intersection* of what is possible using three different schema languages (DTD, W3C Schema, and RELAX NG). In practice, this reduces our modelling requirements quite significantly. It becomes necessary to treat DTD as a sort of poor relation, due to its inherent lack of support for namespaces, or proper data-typing; it also becomes necessary to renounce some of the many additional facilities provided by W3C Schema and RELAX NG for content validation, and to resist the attractions of SGML-specific optimisations such as the ampersand connector — `(a & b)` as a shortcut for `((a,b) | (b,a))`. As currently conceived, the ODD language has to support specification of content models which have the following characteristics:

[2] It was used successfully for the first version of W3C ITS, [6].

[3] `http://www.tei-c.org/Tools/Stylesheets/`

[4] `http://oxgarage.oucs.ox.ac.uk:8080/ege-webclient/` a generic document transformation system which uses TEI as a pivot format. This can manage conversion from (for example) Microsoft Word and Open Office formats to TEI, and from TEI XML to Word, Open Office, HTML, ePub, LaTeX, XSL FO, RDF, JSON etc.

[5] `http://www.tei-c.org/Roma/`

- alternation, repetition, and sequencing of individual elements, element classes, or sub-models (groups of elements) are supported;
- only one kind of mixed content model — the classic `(#PCDATA | g | hi)*` — is 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.

The first item above is of course common to all target schema languages. We address it by the following small incremental changes to the ODD language:

1) *Specification*. We use three pointing elements (<elementRef>, <classRef> and <macroRef>) to refer to components. For example, the old

```
<rng:ref name="model.pLike"/>
```

becomes

```
<classRef key="model.pLike"/>
```

2) *Repeatability*. RELAX NG indicates this by using the grouping <rng:oneOrMore> and <rng:zeroOrMore> elements. We follow the notation (from W3C Schema datatypes) already used in the TEI scheme to express cardinality of attribute values, and replace these by the use of a pair of attributes, *@minOccurs* and *@maxOccurs*, on the elenents <elementRef>, <classRef> and <macroRef>, which give more delicate and consistent control over what is possible within the components of a content model.

3) *Sequence and alternation*. Sequencing and alternation are currently indicated by elements defined in the RELAX NG namespace (<rng:choose>, <rng:group>, etc.) We replace these by similar but more constrained TEI equivalents <sequence> which is like <rng:group> in that its children form a sequence within a content model, and <alternate> which operates like <rng:choose> to supply a number of alternatives.

For handling character data, we follow the W3C Schema approach and define an attribute *@mixed* for each container element.

The other two constraints above are not formally expressed in the current or proposed ODD languages. We consider later some implications of relaxing them in the proposed modification. First, we provide some examples, showing how content models expressed using RELAX NG compact syntax may be re-expressed.

The content for <TEI> is modelled as follows

```
(teiHeader, (model.resourceLike+, \text?)  | \text)
```

in which we have a sequence of a mandatory <teiHeader>, followed by a choice between some members of the 'model.resourceLike' class followed by an optional <text>, or just a mandatory <text>. This is expressed as follows:

```
<sequence>
 <elementRef key="teiHeader"/>
 <alternate>
  <sequence>
   <classRef
     key="model.resourceLike"
     minOccurs="1"
     maxOccurs="unbounded"/>
   <elementRef key="text" minOccurs="0"/>
  </sequence>
  <elementRef key="text"/>
 </alternate>
</sequence>
```

Repetition can be applied at any level. In the content model for <index> `((term, index?)*)` we have a repeated sequence in which the second item is optional. This is expressed as follows:

```
<sequence minOccurs="0" maxOccurs="unbounded">
 <elementRef key="term"/>
 <elementRef key="index" minOccurs="0"/>
</sequence>
```

The content model for <p> is mixed:

```
  (text | model.gLike | model.phrase | model.inter |
model.global | lg)*
```

This is now expressed as follows:

```
<content mixed="true">
 <alternate minOccurs="0" maxOccurs="unbounded">
  <classRef key="model.gLike"/>
  <classRef key="model.phrase"/>
  <classRef key="model.inter"/>
  <classRef key="model.global"/>
  <elementRef key="lg"/>
 </alternate>
</content>
```

With these changes, the ODD language is at least as expressive as it was before, and presents a uniform look to the user. We have lost the ability to use other features of the RELAX NG language, but we regard this as a benefit of the changes. The fact, for instance, that in our old ODD, attributes could be defined using the facilities of ODD (<attDef> and its children) or directly in the RELAX NG content model of elements presented the possibility of considerable confusion.

5. IMPLEMENTATION

Initial implementations of the changes to TEI ODD discussed in this paper have been fairly easy to put in place, and automated conversion of the 500-plus TEI element specifications has been completed. Conversions back to RELAX NG, DTD and W3C Schema are in place, and produce schemas which pass all the previous tests.

The TEI Guidelines serve as a testbed for the ODD language and for the tools implementing it. They consist of a single ODD document managed as a set of over 800 source files in a Sourceforge Subversion repository.[6] Every new release or update of the TEI Guidelines is the result of an integrated workflow in which the whole ODD document is processed to generate schemas and documentation; the document itself, and all examples therein, are tested against the schemas that have just been generated;[7] subset schemas specified by various other ODD specifications are then used to validate a suite of existing test files and the documentation processed to generate readable HTML, PDF, and ePub in one or more languages. The whole process is managed using a pair of Jenkins Continuation Integration servers at different locations.

6. GOING LARGE

As noted above, our current policy is to ensure that the content models expressible in ODD use only the intersection of the features of currently available schema languages. Very few of the features offered by RELAX NG beyond content model definition are used in the TEI. Instead, other mechanisms are used: a <datatype> element is used to associate datatypes with attribute values, using a layer of datatype abstraction additional to that provided by e.g.

[6] https://sourceforge.net/p/tei/code/HEAD/tree/

[7] In some cases, the examples are marked as *feasibly valid* using the felicitous notion introduced by James Clark in Jing, http://jing-trang.googlecode.com/svn/trunk/doc/

W3C Schema; the <valList> element is used to enumerate legal values for attributes, attribute classes, or element content; and the <constraintSpec> feature expresses such niceties as the requirement that a <list> element whose @*type* attribute has the value 'gloss' must contain a sequence of <label> and <item> pairs, or that the @*target* attribute of a <delSpan> element must point to an element which follows the <delSpan> in document order. These constraints are expressed using the ISO Schematron language, though the specification allows for other possibilities. The goal remains to avoid dependence on any one schema language

This pragmatically-motivated and cautious restriction is not however the only possible policy. Removing the intimate dependence of the ODD language on any specific schema language makes it possible to envisage a more adventurous policy, in which we select amongst all the language features on offer to determine those of most relevance to our user community's needs.

For example, the RELAX NG language has a very useful feature called interleave, which constrains an element's content to be a sequence of single specified elements appearing in any order; that is to say, 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 but there is no equivalent feature in the W3C Schema or DTD languages. Suppose now that we add to the ODD language a new grouping element such as <interleave>, or add an attribute @*preserveOrder* taking values 'true' or 'false' to our existing proposed <sequence> element. Generating a RELAX NG schema from such an ODD would be simple; for the other two schema languages, a processor would choose amongst the following strategies, in addition to the normal processing. Firstly, itt can simply reject the construct as infeasible; secondly, it can over-generate; that is, produce code which validates everything that is valid according to the ODD, but also other constructs that are not so valid; or, lastly, it cab generate very lax constraints, but in addition produce Schematron code to catch 'false positives'.

As a second example, consider the need for contextual variation in a content model. For example, a <name> or <persName> appearing inside a 'data-centric' situation, such as a <listPerson> element is unlikely to contain elements such as <del> or <corr>, although these are entirely appropriate and useful when encoding names found on a transcribed inscription. Similarly, 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 permitted within the corpus header — the latter are rather unlikely to include any part of speech tagging for example.

At present ISO Schematron rules allow us to define such contextual rules, and something analogous to them is provided by the W3C Schema notion of base types. Suppose, however, that an XPath-valued @*context* attribute were available on any of the elements <elementRef>, <macroRef>, or <classRef> restricting its applicability. Thus, the content model for <p> might say something like

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

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. Again, we could generate a fully-functional native schema in some languages (W3C Schema), and produce a laxer one enhanced by generated Schematron in others.

Comparisons, as Mrs Malaprop says, are odorous. Each of the three formal schema languages currently available (RELAX NG, W3C and DTD) has distinct adherents and associated software tools. Although it is easy enough to trace their evolution, and to speculate about the differing motivations of their language designers, the practical consequence of this multiplicity is that true interoperability of document grammars becomes more difficult. The fact that these three schema languages diverge so widely in their feature sets suggests that it is prudent to define TEI content models in a way that is fully independent of any of them. Whether that independent definition should take the cautious route of providing only the intersection of their facilities, or the more adventurous one of trying to define a new synthesis, selecting those features most appropriate to the needs of a given user community, is currently less clear. The TEI community, which has long provided a testing ground for advances in document grammars,[8] is perhaps well qualified to decide the question, particularly since (as we have seen) the TEI approach to datatyping already extends the expressive power of the ODD language beyond simple syntactic constraints.

The real challenge of the value of our approach will come in the next decades when, or if, a new formalism replaces XML as the language of choice for structured textual data, and we start work on mapping our TEI abstract language to that new formalism.

7. REFERENCES

[1] Donald E Knuth. Literate Programming. CSLI Lecture Notes 27, Center for the Study of Language and Information, Stanford, California, 1992.

[2] C. F. Goldfarb. *The SGML Handbook*. Oxford, Clarendon Press, 1990.

[3] ISO. *Standard Generalized Markup Language (ISO 8879:1986 SGML)*. International Organization for Standardization, Geneva, 1986.

[4] ISO. *ISO/IEC 19757-2:Amd1 Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG – Amendment 1: Compact Syntax. ISO version of the RELAX NG Compact Syntax*. International Organization for Standardization, Geneva, 2003.

[5] Lou Burnard and Sebastian Rahtz. RelaxNG with Son of ODD. In *Proceedings of Extreme Markup Languages Conference*. Extreme Markup Languages, Montréal, Québec, 2004.

[6] F. Sasaki and C. Lieske. Internationalization Tag Set (ITS) Version 1.0. Recommendation, W3C, 2007. 2010-10-25.

[7] TEI Consortium. TEI P5: Guidelines for Electronic Text Encoding and Interchange. Technical report, TEI Consortium, 2010. 2010-10-25.

[8] It should not be forgotten that the TEI has been a major proponent in the development of what constitutes current orthodoxy in markup languages. The ideas now standardized as XPointer, for example, were first sketched out as TEI extended pointers in TEI P1. Many of the characteristics of the usable subset of SGML which eventually became XML were first explored in the process of defining a scheme adequate to the needs of the TEI community.