The Compromises and Flexibility of TEI Customisation

by James Cummings

Citation

Cummings, James. 'The Compromises and Flexibility of TEI Customisation'. In: Clare Mills, Michael Pidd and Esther Ward. *Proceedings of the Digital Humanities Congress 2012*. Studies in the Digital Humanities. Sheffield: HRI Online Publications, 2014.Available online at: <http://www.hrionline.ac.uk/openbook/chapter/dhc2012-cummings>

Abstract

The ongoing efforts of the Text Encoding Initiative (TEI) have produced a wide-ranging set of Guidelines for encoding textual phenomena. However, standardisation attempts such as this always make compromises in order to achieve their goals. The recommendations of the TEI are no exception to this and this paper examines at some of these compromises along with the benefits and drawbacks they bring. The TEI's extremely flexible method for documenting customisations and extensions will be also considered and how it might benefit interoperability amongst sub-communities of users.

The Compromises and Flexibility of TEI Customisation

by James Cummings

1. Introduction

The Text Encoding Initiative Consortium (TEI) is an international membership consortium whose community and elected representatives collectively develop and maintain the de facto standard for the representation of digital texts for research purposes.[1] The main output the community produces is the TEI Guidelines which provide recommendations for encoding methods for the creation of digital texts. Generally the TEI is used by academic research projects in the humanities, social sciences, and linguistics, but also by publishers, libraries, museums, and individual scholars, for the creation of digital texts for research, teaching, and long-term preservation.

One of the benefits of the TEI's long history is that over many years, experts in very particular aspects of textual phenomena have examined and improved the recommendations as part of the very community it seeks to serve. The TEI takes a generalistic approach to describing textual phenomena consistently across texts of different times, places, languages, genres, cultures, and physical manifestations, but it simultaneously recognises that there are distinct use cases or divergent theoretical traditions which sometimes necessitate fundamentally different underlying data models. Unlike most standards, however, the TEI Guidelines are not a fixed entity as they give projects the ability to customise their use of the TEI - to constrain it by limiting the options available or extending it into areas the TEI has not yet dealt with.

2. TEI Customisation Infrastructure: Modules and Classes

Many who use the TEI Guidelines are unaware that all uses of the TEI are based on a customisation of the fullest possible TEI schema. In most cases this is because the context in which users experience the TEI is either with

an editor which has the fullest schema (*tei_all*) already installed, or in a project where someone else has undertaken the customisation for them.[2] In examining the process of customisation of the overall TEI scheme it must be recognised that it has well over 500 elements for representation of a vast array of textual phenomena. An understanding that the TEI elements are organised into 'modules' and 'classes' helps significantly in the customisation of the TEI. These element definitions are gathered together in modules simply as a way of defining a single group: "A module is … simply a convenient way of grouping together a number of associated element declarations".[3]

Sometimes, as with the TEI's Core module (containing the most common elements) these may be grouped together for practical reasons. However, it is more usual, for example with the 'Dictionaries' module, to group the elements together because they are all semantically-related to one particular sort of text or encoding need. As one would expect, an element can only appear in one module lest there be a conflict when modules are combined.

Almost every chapter of the TEI Guidelines has a corresponding module of elements. In the underlying TEI ODD language (discussed later) both the prose of that chapter of the Guidelines and the specifications for all the elements are stored in one file. It is from this file that both the TEI documentation and the element relationships used to generate a schema are created.

The TEI Class system is similar to this but slightly different: while an element can only appear in one module, it can be a member of many classes. While a module is a single unit, classes can contain not only elements (or attributes) but also other classes or subclasses. Classes are used to express two distinct kinds of commonality among elements. The elements of a class may share some set of attributes, or they may appear in the same locations in a content model. A class is known as an attribute class if its members share attributes, and as a model class if its members appear in the same locations. In either case, an element is said to inherit properties from any classes of which it is a member.[4]

To enable easier comprehension of the many elements that the TEI Guidelines describe, these elements are also categorised into classes on structural or semantic grounds. The primary division of classes is between attribute classes and model classes. In the first of these, all the elements that are members of the same attribute class share the attributes stored in the definition for that class. For example, the class att.internetMedia

contains an attribute @mimeType.[5] There are five members of this attribute class: binaryObject, graphic, equiv, ptr, and ref, which means that each of these elements has a @mimeType attribute. Attribute classes may contain other classes, and attributes from a subclass will inherit the attributes from a superclass which contains that subclass. Indeed, the binaryObject and graphic elements are really a member of the att.media class, which is itself a member of the att.internetMedia class.

Elements which are members of model classes are all allowed to appear in the same place. What this means is that in the construction of the content model of an element it will say what content is allowed inside it. In many cases that element will say members of a particular class of elements are able to be used there. One of the benefits of this slight indirection is that if you want a new element you have created to appear in the same places as an existing element, you simply need to add to it that class. For example, the class model.noteLike is used by many elements (and indeed another model class model.global) to allow things which are note-like to be used inside them. The only members of model.noteLike are note and witDetail.[6] So, in any element content model where model.noteLike is referenced, both note and witDetail are able to be used.

Some of the model classes have the suffix ‘Like’ or ‘Part’ in their name. This delineates two types of groupings. If a model class has ‘Part’ as a suffix, then it is defined by its structural location. For example, members of model.biblPart contain elements which are used inside of the ‘bibl’ element; that is, they are a ‘part’ of that element in the sense of being possible valid children.[7] However, elements with a ‘Like’ suffix are elements that are of similar semantic nature, and thus able to be used at the same point. For example, model.biblLike contains those elements which are ‘like’ the bibl element in that they contain a bibliographic description of some sort.[8] There are other model classes, such as model.inter which do not contain a ‘Like’ or ‘Part’ suffix, and are convenient groupings of elements (often super classes) that all appear in the same place.

Modules and classes are intrinsically related. Most classes are defined initially in the TEI Infrastructure module, and which attributes or elements are available as part of any TEI schema are dependent upon the modules which are loaded. For example, model.phrase contains many subclasses, one of which is model.lPart (for parts of a metrical line such as a caesura or rhyme word). However, if in generating a schema one does not include the Verse module, then the two elements which model.lPart provides, caesura and rhyme, would not appear as an option where the TEI schema uses the

model.phrase class.[9]

Although most classes are defined by the TEI Infrastructure module, a class cannot be populated unless some other specific module is included in a schema, since element declarations are contained by modules. Classes are not declared ‘top down’, but instead gain their members as a consequence of individual elements’ declaration of their membership. The elements declare that they are a member of this special club, and in doing so gain either attributes from or access to areas of the TEI schema. The same class may contain different members, depending on which modules are active. Consequently, the content model of a given element (being expressed in terms of model classes) may differ depending on which modules are active.

3. TEI ODD Customisation

The ability to customise the TEI scheme is something which sets it apart from other international standards. At first glance this may seem contradictory: how can one have a standard that any project is allowed to change? This is because the TEI’s approach to creation of this community-based standard is not to create a fixed entity, but to provide a framework in which projects are able to extend or constrain the scheme itself. They can constrain it by limiting the options available to their project or extend it into areas not yet covered by the TEI. Therefore, it is nonsensical for a project to dismiss use of the TEI because it does not yet have elements specific to its needs. (There are other situations where use of the TEI is not an appropriate choice, but it certainly is not because it doesn’t deal with a particular textual phenomenon.) At the very least any project digitising text should benefit from the long history of the TEI and examine any appropriate recommendations. This ‘customisability’ is both one of the greatest strengths of the TEI approach as well as one of its greatest weaknesses: it is extremely flexible, but it can be a barrier to the interoperability of digital text from sources with different encoding practices.

Each and every project using the TEI Guidelines is already dependent upon some form of customisation even if it is the *tei_all* example customisation with absolutely everything in the TEI Guidelines. For many projects this is enough, but it does projects a disservice if they do not constrain and control the data entry for their project and document it with a TEI customisation.

The concept of customisation originates from a fundamental difference

between the TEI and other standards –TEI tries not to tell users that if they want to be good TEI citizens they must do something this one way and only that way, but while making recommendations it gives projects a framework by which they can do whatever they need to do but document it in a (machine-processable) form that the TEI understands. Such documentation of variance of practice and encoding methods enables real, though necessarily mediated, interchange between complicated textual resources. Moreover, over time a collection of these meta-schema documentation files help to record the changing assumptions and concerns of digital humanities projects more generally.

The TEI method of customisation is written in a TEI format called 'ODD', or 'One Document Does-it-all', because from this one source we can generate multiple outputs such as schemas, localised encoding documentation, and internationalised reference pages in different languages. A TEI ODD file is a method of documenting a project's variance from any particular release of the full TEI Guidelines.[10] The TEI provides a number of methods for users to undertake customisation ranging from intuitive web-based interfaces to authoring TEI ODD files directly.[11] These allow users to remove unwanted modules, classes, elements, and attributes from their schema and redefine how any of those work, or indeed add new ones in a different namespace. One of the benefits of doing this through a meta-schema language like TEI ODD is that these customisations are documented in a machine-processable format which indicates precisely which version of the TEI Guidelines the project was using and how it differed from the full Guidelines.

4. Customisation Case Study: The Stationers' Register Online

A clear example of the use of TEI ODD customisation for the benefit of a research project is the Stationers' Register Online project (SRO). This project received institutional funding from the University of Oxford's Lyell Research Fund to transcribe and digitise the first four volumes of the Arber's edition of the Register of the Stationers' Company.[12] The Register is one of the most important sources for the study of book history in Britain after the books themselves, being the method by which the ownership of texts was claimed, argued, and controlled between 1577 - 1924. This register survives intact in two series which are now at the National Archives and the Stationers' Hall itself.[13]

The pilot SRO has created full-text transcriptions of Edward Arber's 1894 edition of the earliest volumes of the Register (1557—1640) and the Eyre, Rivington, and Plomer 1914 edition (1640—1708).[14] It has also estimated the costs involved in the proofing and correction of the resulting transcription against the manuscript originals, as well as potential costs of transcription of the later series from both manuscript and printed sources. This pilot, a pump-priming project, has produced enough data to demonstrate its usefulness. The intent is to enable future funding bids to digitise the later material as well as implement and launch a useful website that allows researchers to search, browse, and interrogate the complete Stationers' Register, eventually to provide access to corresponding images of the Register if these are made available by the resource holding institutions.[15]

In the case of this initial project, the use and customisation of the TEI recommendations is of particular interest. The keying company chosen has a policy of charging per kilobyte of output produced, yet is content to work to any XML schema, providing it is accompanied by encoding guidelines with examples, and the textual phenomena of the source material to be transcribed and marked up are easily recognisable by keyers. The money saved through creating a byte-reduced, highly abbreviated, and significantly constrained schema based on the TEI, means that the project could also produce transcriptions of the volumes of the Register edited by Eyre, Rivington, and Plomer. However, the focus of this paper (as of the project as it was originally conceived) is the transcription of Arber.

5. The Stationers' Register as a Source

As Arber's nineteenth-century edition of the Stationers' Register existed as a source, it was decided that this was a much better starting point for the pilot than the manuscript materials themselves.[16] That the Stationers' Register is thought of as being a highly complex set of volumes that are difficult to use is precisely what makes them ripe for digitisation. In the earlier volumes the register is also used as a general accounts book for the Stationers' Company, but over time evolves into a more or less formulaic set of entries following a fairly predictable format. As such, it both benefits significantly from being marked up, but the heterogeneity of format poses additional problems in the creation of a consistent markup schema.

In the nineteenth century Arber recognised the potential usefulness of markup and thus marked particular features of the Register surprisingly

consistently in the volumes he edited. The encoding tools at his disposal, however, were only page layout and choice of fonts. The 'nineteenth-century XML', as the presentational markup he chose was termed within the project, was used to indicate basic semantic data categories. The editorial methods he chose are also helpfully documented in his edition's prefatory material, 'On the Present Transcript'.[17] Arber's extremely consistent use of this presentational markup, and the subsequent encoding of it by the data keying company, meant that the project could deduce and generate much of the descriptive markup. If this presentational markup had not existed then a pilot project (with very minimal funding) to produce a digital textual dataset would not have been possible.

As an example of this presentational markup, Arber estimates there are 40,000 names (both of real people and fictitious ones in titles) in his four volumes. Arber recognises these as a potential source of interest (since book history was already an important area of study in his day) and as such differentiates them in various ways. He does note a number of problems inherent in this because members of one livery company might leave it and move to another. Arber says: "the names of all members, whether Freemen or Brethren, of the Stationers' Brotherhood or Company, so far as they could be ascertained, [are set] in a special type technically known as 'Clarendon,'".[18] Names in general are encoded differently. "Names of persons generally, and in the Book Entries of authors real or fictitious, have been placed in Roman Small Capitals".[19] The names of authors are differentiated: "Names of authors used to represent one or all of their works" appear in "Italic Capitals".[20] The care with which Arber constructed his edition benefited the project significantly.[21]

6. Tightening the *tei_corset*

The Bodleian Library's relationship with a number of keying companies meant that the SRO project was able to find one willing to encode the texts in XML to any documented schema. And indeed, very importantly, this particular keying company charged for their work by kilobyte of output. Owing to this, the project realised that it would save money if it could create a byte-reduced schema which resulted in files of smaller size. Such a schema, called *tei_corset* to reflect its constricting nature, replaced the long, human-readable, names of elements, attributes, and their values with highly abbreviated forms. For example, the element became <d>, the @type attribute became @t, and the allowed values for @t were tightly controlled.

This meant that what might be expanded as (18 characters) was coded as <d t="e"> (9 characters). The creation of such a schema was intended solely to reduce the number of characters used in the resulting edited transcription, as an intermediate step in the project's workflow – documents instances matching this schema are not public, since it is the expanded version that will be more useful. This sacrifices the extremely laudable aims of human-readable XML and replaces it with cost-efficient brevity.

The number of elements allowed in the schema was also significantly reduced. While the full TEI scheme has well over 500 elements, only 34 elements in total were allowed in this schema. This meant that the keyers had a reduced choice and so were less likely to make mistakes.

Table 1: Elements included in tei_corset.

TEI Source Module	**Elements Allowed in *tei_corset***
textstructure	body div TEI
figures	cell row table
header	teiHeader
linking	ab seg
transcr	fw space
namesdates	forename surname
core	abbr add cb date foreign gap graphic head hi item label lb list name note num p pb q title unclear

As with all TEI customisations, this was done with a TEI ODD file and as noted above was called *tei_corset* because of the compression being undertaken.[22] This TEI ODD used the technique of inclusion rather than exclusion (that is, it said which elements were allowed instead of taking all of them but deleting the ones it did not want). What this meant was when the project regenerated its schemas or documentation using the TEI Consortium's freely available services, only the original requested elements were included, and new elements that had been added to the TEI since the project created the ODD would be excluded. This part of the TEI ODD file looks like:

Figure 1: <moduleRef/> elements for tei_corset.

```
<schemaSpec ident="tei_corset"
   xmlns:rng="http://relaxng.org/ns/structure/1.0" docLang="en" start="TEI"
   targetLang="en" ns="http://www.tei-c.org/ns/corset/1.0">
   <!-- module references with elements by inclusion -->
   <moduleRef key="tei"/>
   <moduleRef key="textstructure" include="body div TEI"/>
   <moduleRef key="figures" include="cell row table"/>
   <moduleRef key="header" include="teiHeader"/>
   <moduleRef key="linking" include="seg ab"/>
   <moduleRef key="transcr" include="fw space"/>
   <moduleRef key="namesdates" include="forename surname"/>
   <moduleRef key="core"
      include="abbr add cb date foreign gap graphic head hi item label lb list
      name note num p pb q title unclear" />
   <!-- … -->
```

7. Abbreviating Elements

As mentioned above, in the majority of cases the element modifications were quite simple changes to the element's name. The <elementSpec> below, for example, documents the <list> element being renamed to <ls>:

Figure 2: The <list> element in tei_corset ODD.

```
<elementSpec ident="list" mode="change">
   <altIdent>ls</altIdent>
   <equiv filter="corset-acdc.xsl" mimeType="text/xsl" name="list"/>
   <attList>
      <attDef ident="type" mode="change">
         <altIdent>t</altIdent>
      </attDef>
   </attList>
</elementSpec>
```

This <elementSpec> uses the @ident attribute to identify which element it is documenting, and the @mode attribute to record what is happening to it.[23] With the <list> element it is having a 'change' from one name to another and the @mode attribute reflects this. There is an <altIdent> element to provide a new element name: <ls>. The method of changing the @type

attribute to simply be @t is also similar.

This sort of literate programming becomes fairly straightforward once one is used to the concept. However, there is an important additional step here, which is the use of the <equiv> element. This informs any software processing this TEI ODD that a filter for this element exists in a file called 'corset-acdc.xsl' which would revert to, or further document or process, an equivalent notation. In this case a template in that XSLT file transforms any <ls> element back into <list>. This is a trivial renaming XSLT template:

Figure 3: Renaming <ls> to the TEI element <list>.

```
<xsl:template match="ls" xmlns="http://www.tei-c.org/ns/1.0" name="list">
   <list>
      <xsl:apply-templates
         select="@*|node()|comment()|processing-instruction()|text()"/>
   </list>
</xsl:template>
```

In addition to renaming the @type attribute to be @t, some of the other element customisations constrain the values that it is able to contain. For example, in the <n> element (which is a renamed TEI <name> element) the @t attribute has a closed value list enabling only the values of 'per' (personal name), 'pla' (place name), and 'oth' (other name). If this seems counter the vast array of names documented by Arber in his presentational markup, it is because this is captured with the @rend attribute (or its renamed version as @r).

Figure 4: The tei_corset customisation of <name>.

```
<elementSpec ident="name" mode="change">
   <altIdent>n</altIdent>
   <equiv filter="corset-acdc.xsl" mimeType="text/xsl" name="name"/>
   <attList>
      <attDef ident="type" mode="change">
         <altIdent>t</altIdent>
         <valList mode="replace" type="closed">
            <valItem ident="per"><desc>personal name</desc></valItem>
            <valItem ident="pla"><desc>place name</desc></valItem>
            <valItem ident="oth"><desc>other name</desc></valItem>
         </valList>
      </attDef>
   </attList>
</elementSpec>
```

8. Non-Conformant but Conformable

As with many TEI customisations designed solely for internal workflows, the *tei_corset* schema is not in fact TEI Conformant. The popular TEI mass digitisation schema *tei_tite* has the same non-conformancy issues.[24] Both of these schemas make changes which fly in the face of the TEI Abstract Model as expressed in the TEI Guidelines. The *tei_corset,* in addition to temporarily renaming the <TEI> element as <file>, changes the content model of the <teiHeader> element beyond recognition. The element specification for the <teiHeader> element looks like:

Figure 5: The tei_corset customisation of <teiHeader>.

```
<elementSpec ident="teiHeader" mode="change">
   <altIdent>header</altIdent>
   <equiv filter="corset-acdc.xsl" mimeType="text/xsl" name="teiHeader"/>
   <content>
      <rng:ref name="title"/>
      <rng:zeroOrMore>
         <rng:ref name="model.pLike"/>
      </rng:zeroOrMore>
   </content>
   <attList>
      <attDef ident="type" mode="delete"/>
   </attList>
</elementSpec>
```

This documents the renaming of the <teiHeader> element to <header>

which compared to other abbreviations is quite long, but it was only used once per file so had less pressure to be highly abbreviated. The @type attribute is deleted and more importantly the entire content model is fully replaced. This uses embedded Relax NG schema language to say that a <title> element (which is later renamed to <t>) is all that is required, but can have zero or more members of the model.pLike class after it. This enabled the SRO's encoders to put a basic title for the file (to say what edition it was), but gave them nothing but some paragraphs as a place to note any problems of questions they had. This is a significant departure from the metadata-rich version of the teiHeader usually found in TEI documents. The main reason this is unproblematic is that the headers for each of these files were to be replaced with more detailed ones, which were TEI Conformant, at a later stage in the project data workflow.

As an additional method of documenting that these files were not TEI files, the namespace for the schema was changed to one specifically for *tei_corset*. As with all other aspects, this namespace was then reverted back to the TEI namespace when the files were converted back to pure TEI. A number of other non-conformant changes to the TEI abstract model involved the simplification of content models and the allowing of text inside some (usually) empty elements. In the process of up-converting the resulting XML, these were replaced with the correct TEI structures. An example of this is with the customisation of <gap>:

Figure 6: The tei_corset customisation of <gap>.

```
<elementSpec ident="gap" mode="change">
   <equiv filter="corset-acdc.xsl" mimeType="text/xsl" name="gap"/>
   <altIdent>gp</altIdent>
   <!-- allow gap to have text content -->
   <content>
      <rng:text/>
   </content>
   <attList>
      <attDef ident="agent" mode="delete"/>
      <attDef ident="hand" mode="delete"/>
      <attDef ident="reason" mode="delete"/>
   </attList>
</elementSpec>
```

All of these so-called non-Conformant TEI customisations are indeed 'Conformable', that is they are able, through a later processing step, to be reverted or transformed into versions which are purely TEI Conformant.

9. No Struggles with the Class System

In the above customisation of the TEI <gap> element the locally-defined attributes, @agent, @hand, and @reason are removed. In a full *tei_all* schema the <gap> element would have the possibility of many more attributes, but these are provided by its claiming membership in particular TEI attribute classes. For the *tei_corset* schema many TEI classes were simply deleted which meant that the elements that were claiming membership in this class no longer had these elements.

Figure 7: Deleting some TEI attribute classes for tei_corset.

```
<classSpec ident="att.ascribed" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.breaking" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.datable.iso" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.declaring" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.dimensions" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.duration" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.internetMedia" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.naming" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.placement" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.ranging" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.sourced" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.spanning" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.translatable" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.transcriptional" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.divLike" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.editLike" mode="delete" module="tei" type="atts"/>
<classSpec ident="att.segLike" mode="delete" module="tei" type="atts"/>
```

One class which was modified was the att.global class which appears on every element. It had some of its elements (such as @rendition, @xml:base, and @xml:space) deleted while others were modified to be more concise (such as @xml:id being renamed @id and @xml:lang being renamed @lg). Another of these global attributes, @rend, was renamed @r but was crucial to the project's desired markup. As Arber had taken such pains to indicate Stationers vs Non-Stationers it was useful to capture this where the keying company was able to distinguish the font changes. The @rend attribute was provided with an extensive closed list of values because these would capture Arber's presentational markup that would be used to up-convert to semantic XML. A table of @rend values and their equivalent meanings is provided below.

Table 2: @rend (@r) values for tei_corset.

@rend (renamed @r) attribute value	**Meaning in *tei_corset* Schema**
ab	rendered above the line
al	rendered aligned to the left
ar	rendered aligned to the right
b	rendered in bold
bel	rendered below the line
bl	rendered in blackletter font
brl	rendering is bracketed to the left
brr	rendering is bracketed to the right
c	rendered centred
dc	rendered as drop-cap or illuminated initial
f	rendered in a different font
i	rendered in italics
l	rendered on the left
lrg	rendering is of large size
med	rendering is of medium size
n	rendering returns to 'normal'
o	other rendering
r	rendered on the right
rm	rendered in a roman numerals
s	rendered in superscript
sc	rendered in small caps
sig	rendered as a signature
sml	rendered as smaller
st	rendered struck-through
u	rendered in underline

xlrg	rendering is of extra-large size
xxlrg	rendering is of extra-large size

10. Up-Converting Messy Data

The Arber texts are fairly hierarchical and contain clear structural divisions, the most important of which is what the keyers were marking up as <d t=”e”> (or in expanded form a). These divisions contain a single entry for recording one or more works registered with the Stationers Company. Such entries contain fees paid and they are almost always aligned to the right margin by Arber and recorded in roman numerals. The keying company was asked to mark these fees (the <num> element having been renamed to <nm>) and to use the @r attribute to indicate it’s formatting of ‘ar rm’ (aligned to the right and roman numerals). The benefit to the project of them doing this is that it meant that the SRO project could up-convert this simple number into a more complex markup for the fee. The keying company might mark a number as in the image below (in this case it is really for an expense for timber for building work in 1569 rather than an income but makes a better example).[25] This might have been keyed as:

Figure 8: The <nm> element from an SRO file.

```
<nm r="ar rm">vli iijs iiijd</nm>
```

As mentioned above, ‘ar’ means that this was aligned to the right in the original, while ‘rm’ denotes roman numerals. It is precisely because of the keying company getting confused about the use of this (and using it when they shouldn’t have) which means that the data the project has produced in a proof-of-concept pilot will need significant cleaning before being made public.

However, the up-conversion here isn’t simply to revert numbers back to the correct TEI markup, but to up-convert them to even better markup by deriving information from the textual string that is encoded. The tokenisation of the provided amounts into pounds, shillings and pence, and

consistent encoding of the unit indicator as superscript is a key part of this. Arber's edition provided all the markers of pounds/shillings/pence as superscript, so the keying company was not asked to provide it, as the project realised this could be done automatically after the fact and would save more characters. The project also converted the roman numerals to 'arabic' numbers so that easy calculations of total amount of pence (for comparative purposes) could be provided.

To do this, the XSLT stylesheet converted the keyed text string back into a pure TEI and simultaneously broke up the string based on whether it ended with a sign for pounds, shilling, pence, or half-pence. A more complex XSLT function converted the roman numerals in-between these to arabic, and then to pence so that the individual and aggregate amounts could be stored. The markup that results provides significantly more detail than the original input.

Figure 9: The up-converted output as full TEI markup for the same string.

```
<seg type="fee" rend="roman-numerals aligned-right">
   <num type="totalPence" value="1240">
      <!--orig: vli iijs iiijd-->
      <num type="poundsAsPence" value="1200">v<hi rend="superscript"
            >li</hi></num>
      <num type="shillingsAsPence" value="36">iij<hi rend="superscript"
            >s</hi></num>
      <num type="pence" value="4">iiij<hi rend="superscript">d</hi></num>
   </num>
</seg>
```

A TEI <seg> element is used to mark the fee (which is consistent with other elements marking short segments of text in the converted data) with a @type attribute of 'fee' (mistakenly in this case since this example is one of the mistakes found in proofreading that is actually a payment out, not a fee received) and the @rend values are expanded to their human-readable equivalents. As part of the stage in proofreading (and so as not to dispose of the original data in its preservation copy) the original text string that was passed to the conversion function was stored as an XML comment. The <num> element is used, self-nesting, to give the values for pounds, shillings, and pence, with the external <num> giving a 'totalPence' value.[26] It would have been possible to use the <measure> element here if desired to provide a more detailed recording of the information.[27] Up-converting such data is always inexact because of the nature of the large and messy dataset. The

tei_corset customisation helped to limit the options for mis-encoding and control the number of problems the nature of the data caused. This kind of information is beneficial in that it now provides valuable information which can be used for comparative financial study of a significant period of book history. Additionally, being able to find fees where the amount is quite large helped the project find additional locations of encoding errors such as this one in figure nine.

11. Profiting from ODD Customisation

If the SRO project had attempted to generate full-text transcriptions and encoding of the same amount of material directly from the manuscripts it would have had taken several years and significantly more resources to achieve. Working from an out-of-copyright reliable transcription meant that the creation of this data in electronic form was possible. Moreover, the use of TEI ODD customisation significantly benefited the project by reducing the real world cost of output from the keying company. As part of the project funding application, a quick estimate before the digitisation suggested that the project would save approximately 40% in file size from using the *tei_corset* schema. However, when this reverting of markup back to pure TEI was combined with the up-conversion which derived new data and markup, the savings in filesize were well over 60% compared to having asked the keying company to encode this data fully. This additional saving meant the project was able to include additional materials and so added the Eyre, Rivington, and Plomer editions of the Register (1640—1708) as part of the project.

More generally TEI customisation not only documents the encoding that was undertaken by a project but also the intent of the project. It enables projects to control the scope of markup allowed, and thus is a boon when dealing with external suppliers or even multiple local encoders working on the same project. The enforcement of consistency, and the accompanying ability to detect encoding variance was beneficial for the later quality assurance step in the project's workflow.

Another benefit of the documentation of local encoding practice is for the legacy data migration of document instances in the future. Even the conversion of closely related documents such as those from the Early English Books Online - Text Creation Partnership into pure TEI P5 XML can be an onerous task.[28] One proven approach to comparing texts is to define their

formats in an objective meta-schema language such as TEI ODD, and in doing so the precise variation between the categories of markup used is exposed, and more importantly, provided in a machine-processable form.

Such documentation of variance of practice and encoding methods as a TEI ODD meta-schema preserves then helps to enable real, though necessarily mediated, interchange between complicated textual resources. Moreover, over time a collection of these meta-schema documentation files record the changing assumptions and concerns of digital humanities projects.

12. Conclusion: The Unmediated Interoperability Fantasy

One of the misconceptions about the TEI, and indeed any sufficiently complex data format, is that once one uses this format that interoperability problems simply vanish. This is usually not the case. Following the recommendations of the TEI Guidelines does, without question, aid the process of interchange especially when there is a fully documented TEI ODD customisation file. However, interchange is not and should not be confused with true interoperability. I would argue that being able to seamlessly integrate highly complex and changing digital structures from a variety of heterogeneous sources through interoperable methods without either significant conditions or intermediary agents is a deluded fantasy. This is not and should not be the goal of the TEI. And yet, when this is not provided as an off-the-shelf solution some blame the format rather than their use of it. The TEI instead provides the framework for the documentation and simplification of the process of the interchange of texts. If digital resources do seamlessly and unproblematically interoperate with no careful or considered effort then:

- the initial data structures are trivial, limited or of only structural granularity,
- the method of interoperation or combined processing is superficial,
- there has been a loss of intellectual content, or
- the resulting interoperation is not significant.

It should be emphasised that this is not a terrible thing, nor a failing of digital humanities nor any particular data format, but instead this truly is an opportunity. The necessary mediation, investigation, transformation, exploration, analysis, and systems design is the interesting and important

heart of digital humanities.

13. References

Arber, Edward, ed. *A Transcript of the Registers of the Company of Stationers 1554-1640 AD*, 5 vols. London & Birmingham: private printing, 1875.

Cummings, James. 'The materiality of markup and the Text Encoding Initiative', in *Digitizing Medieval and Early Modern Material Culture: Text*. Ed. by Brent Nelson and Melissa Terras, New Technologies in Medieval and Renaissance Studies Series, Medieval and Renaissance Texts and Studies, (Phoenix: Arizona Center for Medieval and Renaissance Studies, Arizona State University, 2012) p. 49-82.

Cummings, James. 'The Text Encoding Initiative and the Study of Literature', in *A Companion to Digital Literary Studies*, Ed. by Susan Schreibman and Ray Siemens, (Oxford: Blackwell, 2008), pp. 451-476. http://www.digitalhumanities.org/companionDLS/.

Eyre, George Edward Briscoe, Charles Robert Rivington, and Henry Robert Plomer, eds. *A Transcript of the Registers of the Worshipful Company of Stationers from 1640-1708*, 3 vols. London: private printing, 1913.

Gadd, Ian Anders. 'Were books different? Locating the Stationers' Company in Civil War London, 1640-1645,' in *Institutional Culture in Early Modern Europe*, edited by Anne Goldgar and Robert Frost. Leiden: Brill, 2004.

Rahtz, Sebastian andJames Cummings. *Kicking and Screaming: Challenges and advantages of bringing TCP texts into line with the Text Encoding Initiative*. In: Bodleian Libraries, University of Oxford, “Revolutionizing Early Modern Studies”? The Early English Books Online Text Creation Partnership in 2012: EEBO-TCP 2012. http://ora.ox.ac.uk/objects/uuid%3Af9667884-220b-4ec9-bb2f-c79044302399 .

TEI Consortium, eds. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. 2.5.0. [26 July 2013]. TEI Consortium. Accessed 6 September 2013. http://www.tei-c.org/Guidelines/P5/.

Footnotes

[1] http://www.tei-c.org/

[2] The latest *tei_all* schema is available at http://www.tei-c.org/release/xml/tei/custom/schema/relaxng/tei_all.rng

[3] For TEI Modules see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ST.html#STMA

[4] For TEI Class System see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ST.html#STEC

[5] For att.internetMedia see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-att.internetMedia.html

[6] For model.noteLike see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-model.noteLike.html

[7] For model.biblPart see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-model.biblPart.html

[8] For model.biblLike see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-model.biblLike.html

[9] For model.phrase see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-model.phrase.html and model.lPart see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-model.lPart.html

[10] For Personalisation and Customisation of the TEI Guidelines see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/USE.html#MD

[11] For example the Roma web interface http://www.tei-c.org/Roma/ or the XSLT stylesheets which underlie this at http://www.tei-c.org/Tools/Stylesheets/

[12] The project PIs were Dr Giles Bergel (University of Oxford) and Professor Ian Gadd (Bath-Spa University) who provided the academic insight, with Pip Willcox (Bodleian Library, University of Oxford) writing the transcription guidelines, liaising with the keying company, and managing the project editorial process. Dr James Cummings provided all the TEI customisation

and up-conversion discussed in this article but is indebted to the whole project team for their support and assistance.

[13] The SRO project was fortunate to get the support of the the Register's custodians: the Worshipful Company of Stationers and Newspaper Makers.

[14] Edward Arber, ed. A Transcript of the Registers of the Company of Stationers 1554-1640 AD , 5 vols. London & Birmingham: private printing, 1875; George Edward Briscoe Eyre, Charles Robert Rivington, and Henry Robert Plomer, eds. A Transcript of the Registers of the Worshipful Company of Stationers from 1640-1708 , 3 vols. London: private printing, 1913.

[15] It is important to note that the scope pilot project was solely the creation of a testbed of data and not the digitisation of the manuscript books nor production of such a Web interface, is within the scope of the pilot project.

[16] There are existing digital facsimiles which also assisted the project. See for example, http://archive.org/details/1913transcriptof01statuoft

[17] Arber, I. 27-30

[18] Arber, I. 27

[19] Arber, I. 28

[20] Arber, I. 29

[21] Another caveat of the projects limitations is to be explicit that it is a digital transcription of these editions, rather than the original manuscripts of the Stationers' Register that the project is creating. While the editors added much (particularly Arber) to the manuscript, they are not fully complete transcriptions of the manuscripts. For example, Arber was only given permission to transcribe those entries which in some way related to books, to members of the Company, and the careers of individual printers, binders, publishers, as well as "dinner-bills 1557—1569, with some other similar items" (Arber, I. 29).

[22] The tei_corset ODD file and corset-acdc.xsl stylesheet are freely and openly available at https://github.com/jamescummings/conluvies/tree/master/tei_corset

[23] For elementSpec see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-elementSpec.html

[24] For TEI Tite see http://www.tei-c.org/release/doc/tei-p5-exemplars/html/tei_tite.doc.html

[25] We have used building work, rather than book title registration, since most entries registering publications at this time were only four pence (recorded variably as ‘ivd ’ and ‘iijd ’) but seeing a record with a larger amount of money (in pounds, shillings and pence) is helpful to demonstrate the complexity of the following up-conversion.

[26] For num see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-num.html

[27] or measure see http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-measure.html

[28] Sebastian Rahtz and James Cummings, Kicking and Screaming: Challenges and advantages of bringing TCP texts into line with the Text Encoding Initiative . In: Bodleian Libraries, University of Oxford, “Revolutionizing Early Modern Studies”? The Early English Books Online Text Creation Partnership in 2012: EEBO-TCP 2012. http://ora.ox.ac.uk/objects/uuid%3Af9667884-220b-4ec9-bb2f-c79044302399