A world of difference: Myths and misconceptions about the TEI

James Cummings
Newcastle University, UK

Abstract

The Guidelines of the Text Encoding Initiative are generally recognized in the digital humanities as important and foundational standards for many types of research in the field. The TEI Guidelines are generalistic, seeking to enable the largest possible user base encoding digital texts for a wide range of purposes. Consulting on many TEI-based projects, teaching TEI workshops, and volunteering as part of the TEI Technical Council, I have encountered many myths, misconceptions, and misunderstandings about the TEI. Indeed, one plenary lecturer once claimed 'the problem with the TEI is it has too many tags and there is no way to change it'. Inspired by myths such as this, this article will detail common misconceptions about the TEI that I have encountered, concentrating on those technical myths that will help increase knowledge about the TEI misconceptions along the way. The article ends with a consideration of why these myths might have arisen, and what might be able to be done about them.

Correspondence:
James Cummings, School of English, Newcastle University, Newcastle Upon Tyne, NE1 7RU, UK.
E-mail:
james.cummings@newcastle.ac.uk

1 Introduction

The Text Encoding Initiative is a mature international consortium of institutions, projects, and individual members. It is a community of users and volunteers that produces a freely available manual of regularly maintained and updated recommendations for encoding digital text: 'The TEI Guidelines'. The TEI and its community have become an important aspect of Digital Humanities for those undertaking digital textual studies. However, the TEI is more than just a community that creates a set of guidelines: in doing so it formalizes a history of the community's concerns for textual distinctions and exemplifies understandings of how to encode them and how these have developed over its existence; it acts as a slowly developing consensus-based method of structuring those distinctions; it is a mechanism for producing customized schemas that reflect an individual project's needs; it produces methods for transformation to and from numerous other formats; and it is a well-documented format for archival long-term preservation.

This article presents some of the myths, misconceptions, and misunderstandings that I have personally encountered while working with TEI research projects and serving on the TEI Technical Council for more than a decade following the release of TEI P5.[1] The myths or misconceptions I will look at include:

- The TEI is XML (and XML is broken or dead)
- The TEI is too big and complex
- The TEI is too simple or general
- There is no way to change the TEI
- You have to be a TEI guru to customize the TEI
- The TEI is too small (or does not have <mySpecialElement>)
- You cannot get from TEI to $myPreferredFormat
- If you use TEI you must learn other technologies
- You cannot do stand-off markup in XML (or TEI)

Digital Scholarship in the Humanities, Vol. 0, No. 0, 2018.
doi:10.1093/llc/fqy071

- XML (and TEI) cannot handle overlapping hierarchies
- There are no tools for the TEI
- Interoperability is impossible with the TEI
- The TEI is only for digital editions, I am doing $otherThing
- The TEI is only for Anglophone or Western works

This list of myths is not meant to be exhaustive, and there are indeed many more, but these seem most important to address for those new to the TEI. I will not 'name and shame' those repeating these misconceptions, since that would be counterproductive. The real intention here is to record these misconceptions, to explode them, and to consider how they might be avoided in future.

2 The TEI Is XML (and XML Is Broken or Dead)

One of the first myths to investigate is the claim that the TEI is XML and XML is broken or dead. The TEI Guidelines were first expressed in SGML as a markup language and only as of TEI P4 moved to recommending XML, but even this recommendation may change in the future. As new languages, technologies, and methodologies for text encoding emerge in future, the TEI Guidelines may move to them or include them as one of a set of ways to serialize digital text, so long as they meet the basic requirements for easy long-term preservation, expressiveness, validation, integration, and mass adoption that is seen with XML. It is important that it is the prose of the TEI Guidelines that is considered normative, not the current markup language they are written in or recommend, nor the schemas generated from them. What is written in the Guidelines in prose is more important than the rules of any generated schema. There are constraints in the prose of the TEI Guidelines (such as honest adherence to the abstract model) which will never be able to be modelled in any schema language, and this is precisely one of the strengths of these community-developed recommendations.

While the TEI Guidelines are currently formulated as XML, it is important to look at the insistence by some that XML is inherently broken or dead as a technology. This claim is often linked with the desire to herald the arrival of new technologies. This does not mean that XML does not have some limitations (the problem of overlapping hierarchies, discussed later, being one often mentioned and the TEI Guidelines have a whole chapter on this), but there are also many solutions to these. While markup theorists may wish to argue the finer points of whether one possible solution to this or that problem is better, most pragmatic projects just wish to undertake their encoding as efficiently as possible. In my experience any inherent limitations, some of which are discussed later, are rarely prohibitive, and projects often prefer to accept the most straightforward solution. There is a natural temptation for people to latch onto the new, especially in computer technology, and to want to dismiss the old, but they do so at a cost. Championing a new format does not necessitate the denigration of existing formats, and they can and do happily co-exist. Moreover, the TEI Guidelines and project encoding concerns are not usually about the format, but instead they are about the rich granularity of information and the use of appropriate technologies for specific tasks.[2]

3 The TEI Is Too Big and Complex

The TEI Guidelines indeed does consist of a large set of recommendations, and there are many levels at which it can be used and understood, which can be daunting. The TEI Guidelines do sometimes enable multiple methods of encoding the same phenomena but strive to do so only to accommodate differing approaches or methodologies. However, the TEI is a modular framework that allows a project, or a subcommunity to choose precisely which elements to make available and where appropriate build processing workflows based on only those elements. The current TEI P5 Guidelines (version 3.4.0) have 569 elements, but no one expects any encoding project to use all of these or necessarily be fully aware of the underlying infrastructure of interlinked classes that form the TEI infrastructure.[3] In

proper use of the TEI, customization is strongly recommended:

> These Guidelines provide an encoding scheme suitable for encoding a very wide range of texts, and capable of supporting a wide variety of applications. For this reason, the TEI scheme supports a variety of different approaches to solving similar problems and also defines a much richer set of elements than is likely to be necessary in any given project. Furthermore, the TEI scheme may be extended in well-defined and documented ways for texts that cannot be conveniently or appropriately encoded using what is provided. For these reasons, it is almost impossible to use the TEI scheme without customizing it in some way.[4]

To address this misunderstanding, it is necessary to provide a brief introduction to TEI customization. Most customization is undertaken by individuals or projects, but in some cases a number of projects band together to use identical or similar customizations of the TEI Guidelines to enable interchange (or even interoperability) between their resources. That is, a community of practice may define a subset of the TEI Guidelines for use within that community, whether the size of that community is one person or thousands worldwide. An excellent example of this is that users of the EpiDoc Guidelines (produced by an international collaborative effort and intended to be used to encode scholarly and educational editions of ancient documents) do not need to learn all of the TEI Guidelines.[5] A more important benefit is that it has enabled the creation of a variety of tools for the publication of EpiDoc resources, and for conversion of texts encoded using the Leiden Conventions to and from EpiDoc. These are conventions often used in print editions of epigraphical or papyrological documents.

Choosing which elements to include or exclude from the final scheme helps to enforce consistency among the encoders on that project—even if that project only has just one encoder. Moreover, the TEI customization file records the information about the customization for future users and is a place to store project-specific encoding documentation. Indeed, the TEI customization file acts not only as a source of documentation but also defines an encoding scheme and may provide recommendations for processing files encoded with that scheme, and it provides a historical record for all of these (documentation, formal encoding scheme, and processing information). Thus a customization file such as this is called an ODD file for 'One Document Does-it-all'. This TEI ODD file acts as a meta-schema source not only for the generation of a schema (to validate your documents) but also for your local encoding manual, customized, and internationalized as necessary for your project. The figure below (Fig. 1) shows a collection of chosen TEI elements documented by a TEI ODD Customization file which is used to generate schemas in RELAX NG (RNG) or XML schema (XSD) format, as well as documentation outputs such as PDF or HTML versions of the local encoding guidelines reflecting the customization undertaken.[6]

The elements of the TEI Guidelines are organized in a modular manner so that those undertaking customization do not need, necessarily, to pick and choose elements on an individual basis. The TEI ODD customization file may include:

- a module as a whole (thus getting all of its elements)
- a module, and ask for only some of its elements (thus getting only those)
- a module, but list those that are not wanted (thus getting any others)

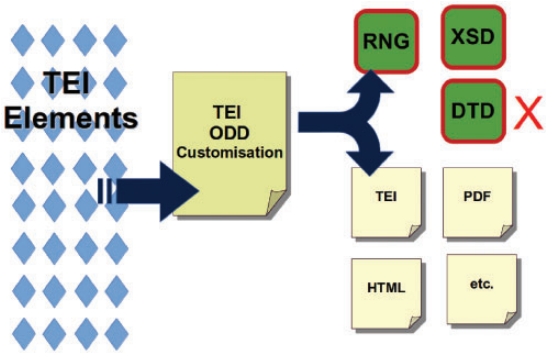

Fig. 1 TEI ODD customization file

Or exclude:

- a module as a whole (thus getting none of its elements)

The figure below (Fig. 2) shows one way of using module-based customization.

There is an important difference between the inclusion and exclusion of particular elements in a TEI ODD Customization. When a TEI ODD file includes some modules and specifies which elements are part of that inclusion, then a schema generated from this will only ever include those elements from those modules. Conversely, if a TEI ODD customization file includes modules but excludes particular elements from that inclusion, then a schema that is regenerated from that TEI ODD file at a future date will automatically include any new elements that the TEI Consortium has added to those modules in the intervening releases of the TEI Guidelines. The choice of approach for a particular encoding project will depend on whether the project wishes to benefit from new elements introduced to that module in the future or remain static with their choices from the beginning.[7] A project's TEI P5 customization is always free to base their customization on any previous version of the TEI Guidelines or they can specify the 'current' version and always get new improvements when regenerating their schemas.

While the TEI Consortium provides Web-based methods for including and excluding elements, the underlying TEI ODD customization file uses the TEI's own vocabulary to record these choices in an XML file. Some TEI customizers choose to edit the XML directly, and in doing so gain more fine-grained control of a customization than using any interface. However, doing this can be complex, and it is not required if the Web-based interfaces suffice. The figure below (Fig. 3) shows the TEI ODD XML required to include a variety of elements.

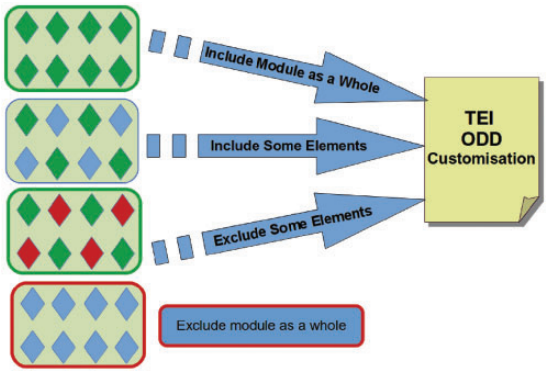

Fig. 2 Including modules in a TEI ODD customization

What this XML tells a TEI ODD processor through these <moduleRef> elements is that it should include the 'tei', 'transcr', and 'tagdocs' modules wholesale.[8] The 'tei' module does not provide any elements but instead is one which instantiates the TEI class structure and should usually be included. TEI modules are usually created in conjunction with a particular chapter of the TEI Guidelines. The 'transcr' module is related to Chapter 11 of the TEI Guidelines 'Representation of Primary Sources'. The 'tagdocs' module is associated with Chapter 22 'Documentation Elements' and provides the elements (such as the <moduleRef> element) which customization files themselves use. The 'core', 'header', and 'textstructure' modules include only specific elements using the @include attribute. This means, for example, if the TEI Guidelines of the future include a new element in the 'textstructure' module, any later outputs generated from this TEI ODD customization file will not know about it. Compare the above figure (Fig. 3) with the figure below (Fig. 4).

The ODD file in Fig. 4 tells any TEI ODD processor that when including the 'textstructure' module, it should include all elements except for those listed in the @except attribute. Currently this would end up with the same list of elements being included, but if the TEI Consortium introduces more elements to the 'textstructure' module in the future, these will appear in the outputs regenerated from this customization.[9] An encoding project need only customize to the necessary degree that assists them in accomplishing (and documenting) their goals, and this kind of flexibility of customization is one of the hallmarks of the TEI. While the TEI is indeed complex, it can be made as small and simple as required.

4 The TEI Is Too Simple or General

In almost complete contradiction to the previous misconception, another criticism often levelled at

```
<moduleRef key="tei"/>
<moduleRef key="core" include="p list item label head author title name"/>
<moduleRef key="header" include="teiHeader fileDesc titleStmt publicationStmt sourceDesc"/>
<moduleRef key="textstructure" include="TEI text body div front back"/>
<moduleRef key="transcr"/>
<moduleRef key="tagdocs"/>
```

Fig. 3 ODD fragment for including elements

```
<moduleRef key="tei"/>
<moduleRef key="core" include="p list item label head author title name"/>
<moduleRef key="header" include="teiHeader fileDesc titleStmt publicationStmt sourceDesc"/>
<moduleRef key="textstructure" except="argument byline closer dateline div1 div2 div3 div4 div5
  div6 div7 docAuthor docDate docEdition docImprint docTitle epigraph floatingText group
  imprimatur opener postscript salute signed titlePage titlePart trailer"/>
<moduleRef key="transcr"/>
<moduleRef key="tagdocs"/>
```

Fig. 4 ODD fragment for including elements with @except attribute

the TEI is that it is 'too simple or too general'. This most frequently comes up when discussing encoding needs with a potential project (preferably but rarely before applying for funding). In my experience this tends to occur when the project claims that the TEI Guidelines are not sufficient to cope with the unique and special research requirements of the particular project. Often this comment is a symptom of a superficial understanding of the scope of the TEI, or a misunderstanding of the generalistic nature of the scheme and the way in which any individual element can be further refined by using attributes or nested levels of encoding.

More than once I have had a conversation with someone with a very basic knowledge of the TEI saying something like 'The <damage> element is far too general, for my research I need something like a <waterDamage> element to say the damage was caused by water!'.[10] One answer is, of course, that one should use the @agent attribute of the <damage> element to specify the damage was caused by water.[11] The possible values of the @agent attribute are not defined by the TEI Guidelines (although some sample suggestions are made, which indeed do not include 'water'), but rather a TEI customization is expected to document its own controlled vocabulary (which may include 'water'). This expectation is expressed by the use of the datatype 'teidata.enumerated' to provide an open list of suggested values in the TEI Guidelines. There are, however, real instances where the TEI is 'too simple' in that it provides only a general level of markup for some phenomena, but this is usually in cases either where there are existing XML standards that the TEI recommends for these or where the TEI community has not pushed the standard forward yet to greater detail in this area. There is an inevitable tension between the desire to add new elements to the TEI scheme to increase its expressivity, and the need to avoid expanding the Guidelines unnecessarily. As a result the TEI strives to recommend related standards where reasonable. An example of this might be the <notatedMusic> element used to encode the presence of music notation in a text. In this case the TEI Guidelines states that 'It is also recommended, when useful, to embed XML-based music notation formats, such as the Music Encoding Initiative format as content of <notatedMusic>. This must be done by means of customization'.[12] While the TEI framework is indeed general, one is able to change the TEI to be as specific as any individual project needs, and while it is simple in some places,

it can be constrained further, expanded, changed, or even mixed with other standards.

5 There Is No Way to Change the TEI

It seems ridiculous, after the previous misconceptions, that anyone would claim that there is no way to change the TEI, and yet, I have sat through an important plenary lecture where the speaker has claimed something like 'the problem with the TEI is it has too many tags and there is no way to change it'.

The TEI Guidelines document a literate programming methodology to customize the TEI framework for specific projects.[13] While there are Web-based tools for creating TEI customizations, in the background they store the information using the TEI ODD XML format. It can take some time to learn the markup needed for TEI customization, but the power over a customization that this gives a project with regard to its encoding, validation, and documentation is well worth the effort—the return on investment for those creating customization is significant.

By using the <elementSpec> element in a TEI ODD XML customization file in the figure above (Fig. 5), this documents a change to the <name> element from the core module. In this case the specifications are missing, but we fill in some of the possibilities in the figures below (Figs 6–8).

The snippet of the customization file depicted in Fig. 6 specifies a change to the specification of the <name> element (hence, the main element is called <elementSpec>). In particular, the gloss of the modified <name> element becomes 'a name', where the original in the TEI Guidelines is 'name, proper noun'. Furthermore, the description of the modified <name> element is changed from the original 'contains a proper noun or noun phrase' to something more jovial.[14] The ODD code in Fig. 6 also replaces the existing class memberships of the <name> element (here commented out). Classes are underlying structures of the TEI framework that elements (or other classes) claim membership of to determine what attributes an element may have (here @type and @subtype from the class 'att.typed') and where it appears in other content models (here anywhere the 'model.nameLike.agent' is allowed).[15] The figures below (Figs 7 and 8) continue where the previous (not well-formed) figure left off.

The TEI ODD customization file documents constraints and potential processing models. The constraints are used to generate additional forms of schema validation (such as Schematron in this instance). Here we have embedded a Schematron rule to report an error when our customized <name> element is used somewhere that is not a descendent of a <p> element. (This is a fabricated pedagogical example, names obviously could appear in many other places, but for the purposes of this customization that would be marked as an error.)

The documentation of processing models is a promising recent addition to the infrastructure of TEI ODD customization. In Fig. 7, the <model> element is used to indicate the intention that when a <name> element that has a @type attribute is being processed for 'web' output, the (intentionally) vaguely specified 'alternate' behaviour should be used. This might change in different forms of Web output or different media of output (such as in print). Here two parameters to the 'alternate' processing behaviour are provided as <param> elements: the 'alternate' parameter and the 'default' parameter. Each has a @value attribute which is an Xpath expression to be interpreted in the context

```
<elementSpec ident="name" module="core" mode="change" xml:id="name-spec">
  <!-- specifications -->
</elementSpec>
```

Fig. 5 ODD <elementSpec> fragment for changing the <name> element

```
<elementSpec ident="name" module="core" mode="change" xml:id="name-spec">
    <!-- Rename the gloss from 'name, proper noun' to just 'a name'. -->
  <gloss xml:lang="en" versionDate="2018-10-25">a name</gloss>

    <!-- For some reason we want to provide a new description.  -->
  <desc xml:lang="en" versionDate="2018-10-25">A name is a proper noun,
    well, maybe also a noun phrase</desc>

    <!-- We've copied the <classes> element over from <name>, but I've
      commented out most of the attribute classes (we could remove them)-->
  <classes mode="replace">
    <memberOf key="model.nameLike.agent"/>
    <!--<memberOf key="att.global"/>-->
    <!--<memberOf key="att.personal"/>-->
    <!--<memberOf key="att.datable"/>-->
    <!--<memberOf key="att.editLike"/>-->
    <memberOf key="att.typed"/>
  </classes>
  <!-- ... -->
```

Fig. 6 ODD <elementSpec> initial fragment for changing gloss, description, and classes

```
  <!-- For some reason we want to provide an additional constraint that names
  should only be used inside paragraphs -->
<constraintSpec scheme="isoschematron" ident="names-in-paragraphs" mode="replace">
  <constraint xmlns:s="http://purl.oclc.org/dsdl/schematron">
    <s:report test="not(ancestor::tei:p)">Error: We only want to allow names inside
      paragraphs.</s:report>
  </constraint>
</constraintSpec>

  <!-- We have added an record of our processing model intention, that this
  should produce an alternate view showing the @type attribute, and be bold. -->
<model behaviour="alternate" output="web" predicate="@type">
  <param name="alternate" value="@type"/>
  <param name="default" value="."/>
  <outputRendition>font-weight:bold</outputRendition>
</model>
```

Fig. 7 ODD fragment documenting constraints and processing models

of each <name> element in a document instance. The first <param> documents that the <name> element's @type attribute should be used for the 'alternate' value, while the second records that the content of the <name> element itself (here '.' in XPath) is to be used as the value for the 'default' parameter. A processing toolchain could take <name> elements that match these conditions and produce Web-based tooltips, or in print output the same 'alternate' behaviour might generate a footnote. Once this basic customization format has been explained, even those editors who barely understand XPath can see that swapping the values would swap which part of the document is used in the tooltip and which in the output text. Developers building infrastructures on top of the processing model have reported that using this and reading the TEI ODD customization file have significantly shrunk and improved their code (Turska et al., 2016).

```
    <!-- We've added a new, very short and unhelpful project-specific example -->
  <exemplum xml:lang="en">
    <egXML xmlns="http://www.tei-c.org//ns/Examples">
      <name type="person">James Cummings</name>
    </egXML>
  </exemplum>
    <!-- Some project specific remarks -->
  <remarks versionDate="2018-10-25" xml:lang="en">
    <p>For our project we have modified the <gi>name</gi> element and allowed it only
      inside paragraphs. The modifications include a new gloss, description, removal of
      several attribute classes, and a new exemplum.</p>
  </remarks>
</elementSpec>
```

Fig. 8 Final ODD <elementSpec> fragment replacing examples and remarks

In the closing fragment of this <elementSpec> (Fig. 8), the examples and remarks of the <name> element are modified using the <exemplum> and <remarks> elements. These elements are very useful for projects that wish to provide examples which match the project's materials and additional element-specific encoding notes for the documentation that they will generate from this TEI ODD.

A common practice that the example in the figures above does not include is the modification of attributes and lists of their possible values. To record that a project is changing the values of an attribute involves a number of steps inside an <attList> element. In the example below (Fig. 9), a TEI ODD customization changes the @agent attribute of the <damage> element of the 'transcr' module (on the representation of primary sources). Inside a list of attributes (using the <attList> element), a single attribute definition (using the <attDef> element) is changed by providing a replacement list of values (using the <valList> element) detailed with the <valItem> element.

In this case the attribute values are in a list of 'semi' using the @type attribute on the <valList> element, which means that these should be considered as strongly suggested values to encourage standardization but one can include new values where necessary. If a value of 'closed' had been used instead, schemas generated from this customization file would allow only the listed values.

A TEI ODD customization file is also able to contain as much prose description and other local encoding information as needed by the project. Element specifications such as these are usually stored inside a <schemaSpec> element but can also be stored elsewhere in the TEI ODD customization file and referenced from within the <schemaSpec> element. Indeed, the <elementSpec> elements in the figures in this section all have @xml:id attributes which would enable them to be embedded alongside the prose and pointed to from the schema specification using a <specGrpRef>. (Though more properly these <elementSpec> elements would themselves be embedded inside a <specGrp> element.) The benefits of doing this should not be underestimated, for example, this enables documentation writers to situate the change in the schema precisely at the point where this change is being documented in prose. If later a project wanted to update the available values for the @agent attribute on the <damage> element (perhaps to include 'fire' alongside 'water'), they could change the <valList> above and any relevant accompanying prose at the same point in the documentation.

The TEI customization demonstrated in this section shows conclusively that one can indeed change the TEI. However, this has demonstrated the more complex TEI ODD XML format which not all TEI users may wish to grapple with and raises the question of whether you need to be some sort of guru to customize the TEI.

6 You Have to Be a TEI Guru to Customize the TEI

There are those that feel the TEI can only be controlled by some elite priesthood, but this is simply not the case. While it is true that some of the

```
<elementSpec ident="damage" module="transcr" mode="change" xml:id="damage-spec">
   <attList>
     <attDef ident="agent" mode="change">
       <valList mode="replace" type="semi">
         <valItem ident="blot">
           <desc>There is an ink blot</desc>
         </valItem>
         <valItem ident="fading">
           <desc>It is faded</desc>
         </valItem>
         <valItem ident="hole">
           <desc>There is a hole</desc>
         </valItem>
         <valItem ident="overwriting">
           <desc>There is overwriting</desc>
         </valItem>
         <!-- ... -->
         <valItem ident="water">
           <desc>There is water damage</desc>
         </valItem>
         <valItem ident="other">
           <desc>There was some other agent of damage</desc>
         </valItem>
       </valList>
     </attDef>
   </attList>
</elementSpec>
```

Fig. 9 Providing a list of values for the @agent attribute of the <damage> element

methods for customizing the TEI, such as those shown above, can be intimidating at first, they still can be mastered with a little practice. Moreover, the TEI Consortium provides Web-based tools to create TEI customizations. At time of writing, most customizers use the Web-based Roma tool provided by the TEI Consortium (http://roma.tei-c.org/); however, since this is showing its age, a new tool is being created.[16]

In Fig. 10, the Roma interface is being used to remove the <analytic>, <biblStruct>, and <binaryObject> elements from this particular customization. Tools such as this allow users to browse through the options available from the TEI framework and select those modules or elements that fit their needs. The customization undertaken by a project can start from scratch or could use a number of existing TEI customizations as starting points.

The overall quality of the customization, regardless of the method by which it is created, does depend somewhat on the knowledge the customizer has of the TEI Guidelines. However, customizations may be modified, updated, and evolved over the lifetime of an encoding project as the needs of the project (and TEI experience of the customizer) change. Those new to customizing the TEI are encouraged to discuss any concerns or questions they may have openly on the TEI's mailing list. Moreover, one project may have different customizations (and thus schemas and documentation) for different stages in their workflows. For example, they might have one for initial data entry, a different one for proofreading, and a tighter one for pre-publication validation.

Although it would be difficult for current or future Web interfaces for editing TEI ODD customization files to allow for the full expressivity possible in the TEI ODD vocabulary, it is certain that these will provide tools for the most common tasks in customizing the TEI. The existence of such tools and their support by the TEI Technical Council or the TEI community means that one does not need to be a TEI guru to customize the TEI.

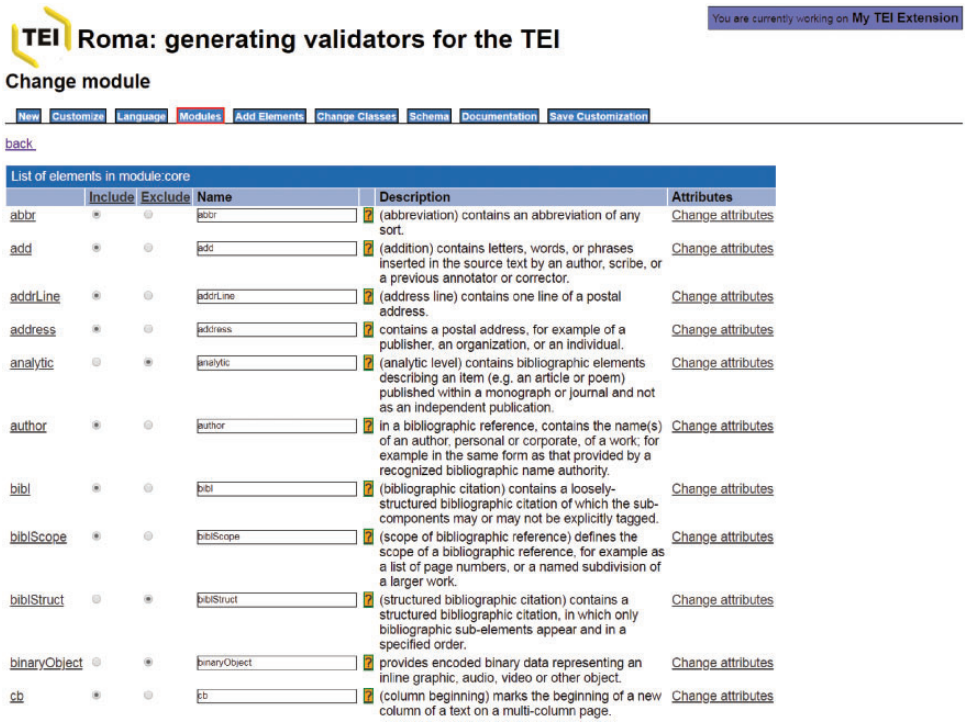

Fig. 10 The (dated) TEI Roma Web interface: http://roma.tei-c.org/

7 The TEI Is Too Small (or Does Not Have <mySpecialElement>)

When someone argues that the TEI Guidelines are too small what they usually mean is that the TEI Guidelines do not include a special element whose mere name implies that its existence might make their particular encoding easier. Occasionally they might really mean that the TEI does not yet have elements for a particular form of encoding. In some instances, the TEI provides a general solution where others might prefer more specific markup. For example, while the <msDesc> element for describing manuscripts can also be used for early-printed books, some are reluctant to do so simply because of the name and its development for manuscript cataloguing. Moreover, there is currently no general-purpose <object> element for the description of non-manuscript or print objects in as much detail.[17] While there is a <collation> element (for describing how the leaves of a manuscript are physically arranged), this is suitable for a prose description or compiled collation formula rather than a more structured record of the information.[18]

Both of these are ways in which the TEI is too 'small', not having an <object> element and only having a limited <collation> element. However, what makes the TEI different from most other standards is that any user may add new elements to it and do so in a manner (using the TEI ODD customization format) which fully integrates it into the TEI infrastructure and acts as documentation for how their project's outputs vary from the version of the TEI that they are using. This also acts as a useful method to suggest changes to the TEI by providing them a copy of a new customization.[19] If a user really wants <mySpecialElement> to be part of the TEI Guidelines, there is a straightforward, open, community-based process for proposing that.[20] The figure below (Fig. 11) demonstrates how to add an entirely new element, in this case a <specialName>.

In this case the <elementSpec> element documents the creation of a <specialName> element, in

```
<elementSpec ident="specialName" mode="add" prefix="jc_" xml:lang="en"
  ns="http://james.blushingbunny.net/ns.html" xml:id="specialName-spec">
  <gloss xml:lang="en">Special Name</gloss>
  <desc xml:lang="en">A "special" type of name created for our ODD</desc>
  <classes>
    <memberOf key="att.global"/>
    <memberOf key="model.nameLike.agent"/>
    <memberOf key="att.personal"/>
    <memberOf key="att.datable"/>
    <memberOf key="att.editLike"/>
    <memberOf key="att.typed"/>
  </classes>
  <!-- giving a new content model -->
  <content><textNode/></content>
</elementSpec>
```

Fig. 11 Creating an entirely new name element <specialName>

a new namespace that is not the usual TEI namespace. Any new elements that customizations create should not claim they are in the TEI namespace and instead provide a new one. This <elementSpec> provides a gloss and description, makes it a member of the same TEI classes as the <name> element, and allows only text nodes as its content model.[21] Any resulting documentation or schemas generated would allow this element in the same places that <name> elements are allowed.

8 You Cannot Get from TEI to $myPreferredFormat

There are many reasons why someone might want to process TEI P5 XML files into other formats. The TEI Guidelines currently recommend encoding texts in XML, which is an easily processable markup language with libraries for handling it in dozens of programming languages. Other simple scripting languages like XSLT and XQuery exist specifically for transforming and querying XML structures. It is very unlikely that it is impossible to programmatically get from the TEI to any other format, though in some cases the cost of doing so may be prohibitive. At time of writing, the TEI Consortium provides best-effort XSLT stylesheets for transformations to or from around forty other formats.[22] Many of these conversions are also available via the TEI Consortium's OxGarage convertor.[23] It is impossible for the TEI Consortium to provide conversions to every known format, nor is this desirable. In most cases the target format will be specific to the needs of a particular project and maintaining such conversions should necessarily be the responsibility of that project (or sub-community) rather than the elected volunteers of the TEI Technical Council who are busy dealing with bugs and feature requests for the TEI Guidelines, maintaining the infrastructure for the TEI Guidelines, and associated software.[24]

From a project point of view, the important thing with the creation of a data model is ensuring the granularity of information to properly express the project's understanding of the phenomena it is encoding. In some cases, it may be better to encode materials in TEI and convert to a format necessary for a later stage in a workflow, for example for an analysis or publication stage. In other cases, the TEI might not be as useful an encoding format as the desired format, and TEI might simply be exported later for long-term preservation. If the other format fits better with the research being undertaken then the solution is simple: use that format! It is more important to do the research enabled by your creation of digital text than it is to use any particular open international standard, however, reinventing the wheel should always be avoided if possible. If your preferred format is an XML vocabulary, then, if desired, a project could even use the TEI ODD

customization language to document this vocabulary as the ODD language 'is not tied to the TEI and can be used to define schemas for any XML language' (Rahtz and Burnard, 2013). There will always be the possibility of converting from the chosen format to the TEI for long-term preservation. But where the TEI is truly not suitable, then it is more important to use the appropriate technology for that research.

This does not mean that the conversion between formats is unproblematic or always simple. Indeed, it is rare that there is not some complication in moving between any two formats, since they may encode semantics that are significantly different, or information at different levels of granularity. Creating careful conversions is a skill and limitation of resources available to a project will determine how much work may be put towards this sort of work, and up-converting data to add greater richness of markup will always be more difficult than so-called 'lossy' conversions. Such 'lossy' conversions may not always preserve distinctions, granularity, or semantics in the output format. This is not always a bad thing. For example, if a piece of research software only takes data in a less-granular format, it is common to convert a copy to that less-rich format on which to run the analysis while keeping the original as the 'real' copy of the data. That research software might be used for the publication or analysis of the data while still retaining the richer version of the data for long-term preservation enables other processes to undertake different analysis in the future. In many cases projects use TEI as their base format and convert to a number of different formats for publication and analysis as needed. In other cases, projects store their data in a different format and create an export to TEI to aid interchange. Both of these are completely reasonable ways to use the TEI and help to demonstrate that one usually can get to or from the TEI to other formats.

9 If You Use TEI You Must Learn Other Technologies

One barrier to using the TEI for some is the assumption that, if they create resources following the recommendations of the TEI Guidelines, they must then learn all of the other related technologies for the processing, display, and publication of XML. This is a misunderstanding that rejects the notion of modern research projects as collaborative enterprises or did not factor in the real economic costs for doing so in their funding model. While there are those digital scholars who will master both the TEI Guidelines and a variety of tools for analysis and querying of TEI resources, there are others who can use technologies to transform and manipulate these resources to a researcher's specifications with only a minimal knowledge of the TEI itself. It is often better to collaborate with technical developers who have the requisite skills needed to produce interfaces which can help answer the research questions at hand.

While it is true that some of those using TEI also go on to learn other XML technologies, this is because of the power that it gives them for publication and analysis. If an encoder is not involved in those aspects of a project, they do not need to learn other technologies. However, when asked by researchers familiar with the TEI but who do not know any programming languages, I have often recommended that they start with related technologies like XPath, XSLT, and XQuery depending on their needs. Indeed, I would recommend at least a very basic knowledge of XPath for encoders working on large XML text collections, especially if they are proofreading or revising them, because of the power it gives them in locating structures and inconsistencies in the marked-up text. XPath also has the benefit of being a fundamental component of other technologies such as XSLT and XQuery, so the few hours of experimentation with it that will enable someone to become effectively proficient in XPath can also be leveraged if the researcher eventually decides to learn more. That said, there is no requirement for those using TEI to learn these technologies or other software. There are many projects where the editorial acts of encoding texts are kept completely separate from those undertaking the analysis or publication of them.

Increasingly there are also general purpose tools like TEI Boilerplate or CETEIcean for lightweight publication, TEI Publisher built on top of eXist-

db, as well as more specific technologies such as the Edition Visualization Technology for those producing that form of critical digital edition.[25] The TEI Consortium provides stylesheets for conversion to/from the TEI, as mentioned earlier, and these are also available for use in editors such as the oXygen XML Editor.[26] The TEI Archiving, Publishing, and Access Service (TAPAS) is a project that enables TEI users an easy method to publish their TEI files without infrastructure of their own.[27]

One of the more recent introductions to the TEI Guidelines, the ability to document intended processing models inside a TEI ODD customization file, gives developers a method to generate software based on implementation-agnostic instructions stored in the customization file. The eXist-db TEI Publisher makes good use of this to enable those without development skills to modify the display of their editions through changes to the TEI ODD customization. While it might significantly increase the benefit one can derive from TEI resources, it is not necessary to learn other technologies to make use of the TEI. The modern nature of research collaboration and generalized tool development are at least two reasons why one does not necessarily need to learn other technologies.

10 You Cannot Do Stand-off Markup in XML (or TEI)

This myth displays not only a misunderstanding of XML but also an unfamiliarity with the TEI. While many encoding practices are done using embedded or inline markup (and indeed that was how most XML was originally conceived), it is also possible, and indeed increasingly common, to create resources as a set of interlinked files, often encoding information in a stand-off or out-of-line method.[28] Using a variety of techniques, any XML document can point to fragments or even individual characters of other parts of itself or other local or remote documents.

The TEI Guidelines have a variety of elements and attributes specifically for use with stand-off and fragmentary structures (such as the <link> and <join> elements, and the @part attribute).[29] Other examples might include the ability to store as out-of-line markup a critical apparatus entry of variant readings (using the <app> element). This <app> element may be stored completely separate from a flatter textual edition which is pointed into from the <app> element.[30]

One of the major changes in TEI P5 is the move of many attributes to take URI-based values, which means they can point both internally and externally to the TEI document. With the subsequent introduction of the <prefixDef> element, encoders are now able to document private URI syntaxes making such pointing attributes even easier to use.

While you can certainly do stand-off markup in the current version of the TEI, there is still room for improvement. Piotr Bański notes some of these limitations of stand-off markup within the TEI framework and has been working to improve this (Bański, 2010). For example, there are significantly advanced proposals for additional elements for embedding stand-off markup and linked data. What is really needed is more and better documentation on using these features of the TEI, and better user-friendly general tools for creating and processing stand-off markup.[31] While there is much more that could be said about stand-off and out-of-line markup versus embedded markup, and how the TEI should deal with this better, it should be evident that the TEI Guidelines do provide the ability to handle stand-off markup.

11 XML (and TEI) Cannot Handle Overlapping Hierarchies

This is an often-touted problem by developers seeking to find problems with XML and replace it either with a new technology or sometimes a different old technology with which they are more familiar. When fear of XML technology is the cause of their objection, then this should be treated with scepticism. That XML has difficulty with overlapping hierarchies is not, in itself, strictly a myth. This concern, indeed, pre-dates XML and existed as a problem noted in SGML (Renear et al., 1996). However, it is usually a misunderstanding that assumes that, since there is a limitation in one area, a whole

standard should be abandoned. While it is true that XML as a data structure limits embedded expressions to a single hierarchy, it has a variety of solutions for this built in, such as the use of empty elements as boundary markers for other hierarchies and URI-based pointing for stand-off or out-of-line markup. This is not some secret or new problem, the TEI Guidelines contain a whole chapter on Non-hierarchical Structures which documents some of the more popular solutions.[32] Others are embedded throughout the TEI Guidelines, and users of the TEI generally follow Renear's suggestion that analytical perspectives often determine the hierarchies one uses, and where non-hierarchical perspectives exist these can often be represented through hierarchical sub-perspectives (Renear et al., 1996). For example, while there is often a conflict between the intellectual structures of paragraphs with the physical structure of pages, the TEI Guidelines recommend the <pb/> (page beginning) element as an empty element to record page breaks. To some users, not being able to enclose both of these simultaneously in elements which wrap around them with start and end tags appears to be a major limitation, while almost all the projects I have been involved with did not find this a significant problem at all. The concern of overlapping hierarchies in embedded markup may be felt more keenly by markup theorists (DeRose, 2004). Some in the markup community long for a more elegant markup system that does not have this limitation, whereas pragmatists tend to doubt that the proposed solutions will achieve XML's wide-spread adoption and benefits.

If one is alternating between two set hierarchies (for example physical and intellectual), then there are a variety of XSLT stylesheets which can transform overlapping hierarchies such as these to swap to the other hierarchy.[33] That in most cases XML only represents overlapping hierarchies in oblique methods proves to be almost no problem for pragmatic projects. Most are happy to prioritize one hierarchy (usually the intellectual) over others (such as the physical) or move to a basic out-of-line markup solution. Those projects for which it is a significant concern are also those likely to be capable of implementing the various mitigating techniques proposed over the years.[34] Given solutions such as these, and the ability of the TEI to use stand-off or out-of-line markup, it is reasonable to claim that the TEI can handle overlapping hierarchies.

12 There Are No Tools for the TEI

As the TEI is currently formulated in XML, and there are thousands of tools which understand XML, this claim is obviously nonsensical at one level. However, those who believe this myth often mean something slightly different: they will claim that there are no tools that understand the TEI vocabulary itself and leverage this to produce the output they desire. The many TEI Consortium Stylesheets have already been mentioned along with TEI Publisher, TEI Boilerplate, and CETEIcean, so clearly this myth is also false. This is not to say that more tool development would not be beneficial.

The figure above (Fig. 12) shows a screenshot of a fragment of the TEI Consortium Wiki listing tools placed in the 'Tools' category. Not all of these tools are for use with only TEI files nor is this in any way an exhaustive list of tools. More accurately these are the tools that people have chosen to list on the TEI Consortium Wiki and so are a very small subset of what is actually available. Most tools that projects create are for their own bespoke purposes rather than generalized tools for any TEI document. Sadly, even many of these are not openly released. There are both open source and commercial XML editors, such as the oXygen XML Editor, which have built-in support for the TEI.

13 Interoperability Is Impossible with the TEI

This misunderstanding has sometimes been given as a justification of why a particular project might prefer to use its own bespoke format rather than a better known format like the TEI. The idea suggested is that as TEI use is so variable, it is near impossible for resources from different TEI customizations to interoperate with each other, thus they might as well reinvent the wheel. The TEI

Pages in category "Tools"

The following 132 pages are in this category, out of 132 total.

A
- ANGLES
- Abbot
- Anthologize
- Apache Cocoon

B
- BBEdit
- Base-X
- Berkeley DB XML
- BibIStruct-to-RIS
- Byzantium

C
- CATMA
- CETEIcean
- CWRC-Writer
- Classical Text Editor
- Cocoon epub Compiler
- CollateX
- ConTeXt
- CoreBuilder
- CorpusReader

D
- DC-dot
- DHConvalidator
- DHWriter
- DLXS
- Data Dictionary Generator (DDG)
- Digitization tools
- Diple
- Doctored.js
- Docx2tei
- Drupal

E
- EATS
- EHumanities Desktop
- EPPT
- ETDPub
- EVT
- EXist
- Ediarum
- Edition Visualization Technology
- Editix
- Editors
- EpiDoc stylesheets
- Epub-tools
- Exchanger XML Editor
- Exmaralda

F
- FSD Validator
- Fedora

F cont.
- FontoXML
- FromThePage

G
- GROBID
- Geany
- GutenbergToTei.py

H
- HOCR2TEI
- How to validate an XML document against a Schematron file

I
- ImageMarkupTool
- IslandoraTEIEditor

J
- JEdit
- JuxtaCommons
- JuxtaWS

K
- Kernow
- Kiln

L
- LombardPress print style sheet converter tool

M
- Serving "application/tei+xml" from Cocoon
- TEI media type
- MarkLogic Server
- Morphadorner

N
- Nidaba

O
- OXygen
- Oddbyexample
- Odt2tei
- OxGarage

P
- P4toP5
- PG2TEI
- Pandoc
- PhiloLogic
- PoetryVisualizationTool
- Pub2TEI

R
- RefDB
- Roma

S
- SDPublisher
- Sacodeyl Annotator
- Serna
- Sxedit

T
- TAPAS
- TAPoR
- TEI Boilerplate
- TEI Critical Apparatus Toolbox
- TEI LookUp
- TEI Publisher
- TEI Transviewer
- TEI Web Editor

T cont.
- TEI-emacs
- TEICHI
- TEIDebian
- TEILiteEditor
- TEITOK
- TEITags
- TEIViewer
- TEIZoteroTranslator
- TILE
- TLex
- TXM
- Tei-xsl
- Tei2html Stylesheets
- Tei2wikitext
- TeiDisplay
- TeiPublisher
- Teipub
- TextMate
- Textual Communities
- Thutmose II
- TokenX
- Transpect

U
- Upconversion

V
- VLE
- Versioning Machine
- Vesta

W
- WebLicht
- Wed
- Wiki2TEI
- WordHoard

X
- XML Copy Editor
- XMLSpy
- XMLStarlet
- XMLmind
- XPAT
- XQuery
- XSLT
- XSLTdoc
- XTF
- Xaira
- Xeditor
- Xmllint
- Xproc-ebook-conv

Z
- ZoteroToTEI

Fig. 12 A snapshot of the Tools category from the TEI Consortium wiki[44]

Guidelines are different from many static standards because they accept the inevitable truth that projects using it need the ability to customize, constrain, and extend the standard they are using. A conflict will always exist between the expressive freedom the TEI gives any individual project and the notion of seamless interoperability with others (Bauman, 2011). The TEI Guidelines provide a mechanism (the TEI ODD customization format discussed earlier) for documenting any local project changes to the standard. The more documentation (human and machine-readable) is provided, the more likely any barriers to interchange and interoperability can be overcome. The existence of TEI customization does mean that two TEI projects might be using very different views of the TEI framework, and this inevitably leads to a tension between the freedom of customization and the fragmentation of the standard within the community (Cummings, 2008). However, that does not imply that interoperability (or at least interchange) is impossible, merely that it can be difficult depending on the degree of differences between the schemas. Luckily, these differences should be recorded in the TEI ODD customization which can be programmatically compared (Burnard and Rahtz, 2004). The TEI Guidelines also include a notion of 'TEI Conformance' that one should think would help with interoperability.[35] While these rules do help in setting some basic ground rules for what constitutes a TEI document, the real challenge for interoperability is the variation between completely valid and conformant TEI documents.

However, seamless interoperability is not what the TEI Guidelines are meant to achieve, rather the possibility for the interchange of texts (Unsworth, 2011). The difference between interchange, one of the original goals of the TEI, and interoperability mostly relies on the degree of human intervention required in the mediation between formats. The degree of interoperability needed between two resources also depends on whether this is bidirectional interoperability (both resources using each other's materials) or more commonly unidirectional. In my view the latter really becomes a form of mediated interchange regardless of whether that is 'negotiated' or 'blind'.[36] Elsewhere I have argued that the notion that unmediated interoperability between such varied resources should be automatic is a deluded fantasy, as there will always be a necessary step of mediation between the resources (Cummings, 2014). This means someone (or some software program) needs to understand the differing formats, perhaps through analysis of the TEI ODD customization. The solution to the problem is proper (machine-readable) documentation, which expresses the differences between each of the projects and the full TEI. While programmatically reading these two customization files will identify where the differences are, there is always a human element in determining how to handle them. When trying to interchange resources, projects often find it useful to downgrade their richer local copies to common TEI subsets (such as TEI Lite or TEI simplePrint). This also has many other benefits. As Martin Holmes notes in reference to downsampling their materials to less-rich TEI formats, this 'constitutes what we might call ''enacted documentation''' forcing them to reflect on the complexities of their encoding choices (Holmes, 2017). Holmes has also created a 'CodeSharing API' which gives an interoperable method for sampling the encoding practices of a project using it.[37]

Interoperability will always remain a problem for any standard with a significant degree of expressivity in the creation of document instances. However, it is much easier to convert between the outputs of two well-documented projects (with both prose and schema specifications in a TEI ODD) than it is between documents from projects following their own bespoke markup systems. Experience shows that the benefit of a shared adherence to a general underlying framework, even when using it to different ends, far outweighs the difficulty in disentangling bespoke systems of encoding.

14 The TEI Is Only for Digital Editions, I Am Doing $otherThing

One of the popular uses for the recommendations of the TEI Guidelines is for the production of digital editions. However, it is also used for the creation of many other resources. For example, while the

recommendations in Chapter 10 'Manuscript Description' are useful when properly describing a manuscript as metadata for a digital edition, they are also used by libraries for fully detailed manuscript catalogues.[38] There are also modules in the TEI for dictionaries, linguistic corpora, and graphs, networks, and trees.

When creating output from a TEI-encoded file, there is a natural assumption by some that there is a one-to-one relationship between this file and a Web-based 'digital edition' as output. However, that is a problematic way in which to view and undertake TEI encoding (Turska, Cummings, and Rahtz, 2016). If one is using the recommendations of the TEI properly, then it is possible to create so much more than a single edition view of this output. One can generate not only multiple editions but also such outputs as supplementary files, indices, databases, interactive visualizations, glossaries, and camera-ready print copy, amongst many other possibilities. TEI is used for creating born-digital ancillary resources as well, such as bibliographies, working papers, meeting minutes, and slides for lectures, as well as other teaching materials. Some research projects create large TEI resources which are never intended for a one-to-one Web publication, but as text-bases for querying and analysis. However, if there is a good, open, international standard for doing '$otherThing' (whatever that might be), then it may be a better idea to use that standard if TEI does not cater for a project's specific needs, but it is vital to make sure that really is the case by consulting the TEI Guidelines and its community. Even though the TEI Guidelines are for many more things than digital editions, the choice of standards to follow should be based on using the appropriate formats for that particular research, in full knowledge of the benefits and drawbacks of the possible formats.

15 The TEI Is Only for Anglophone or Western Works

The TEI Guidelines strive to be applicable to encoding any form of text, from any time period, in any language and writing system. While the TEI has indubitably arisen from a western context, the community strives to broaden the scope of its examples, to extend the coverage of various forms of non-western textual phenomena, to improve its internationalization and localization mechanisms, and to deliberately expand the diversity of the community itself.[39] Those encoding documents following the TEI Guidelines use Unicode to represent characters in digital form. However, not all characters for all human languages for all times are already available in Unicode, so the TEI Guidelines also have built-in recommendations for describing non-Unicode characters as well as recording scribal glyph-variants.

The TEI has mechanisms for internationalization and localization which enable both the TEI Guidelines and individual project customizations to provide descriptions of its markup in any desired language. The TEI Guidelines are written in English and have generally not been translated as a whole, but for a selection of languages where volunteers have undertaken the work, the glosses and descriptions of elements, attributes, and attribute values have been translated. This means that these can be displayed with in the convention outputs of the TEI Guidelines (Web pages, epub, pdf, etc.), or even from the generated schemas as tooltips or pop-ups in some XML editors, in the encoder's preferred languages. Currently, the many of glosses and descriptions have been translated into Chinese (Taiwanese), French, German, Italian, Japanese, Korean, and Spanish. While the TEI has the infrastructure in its TEI ODD customization to enable users to rename objects like elements themselves, users report that encoders that are not fluent in English are not resistant to the (mostly English-derived) element names as long as the descriptions of them are in their preferred language.

Although these mechanisms are a good start, there is always room for improvement, but this requires volunteers fluent in the language willing to donate a significant amount of time for translation. The figure below (Fig. 13) shows the reference page for the <abbr> element in Chinese, demonstrating the kind of benefits given to those encoding in a different language.

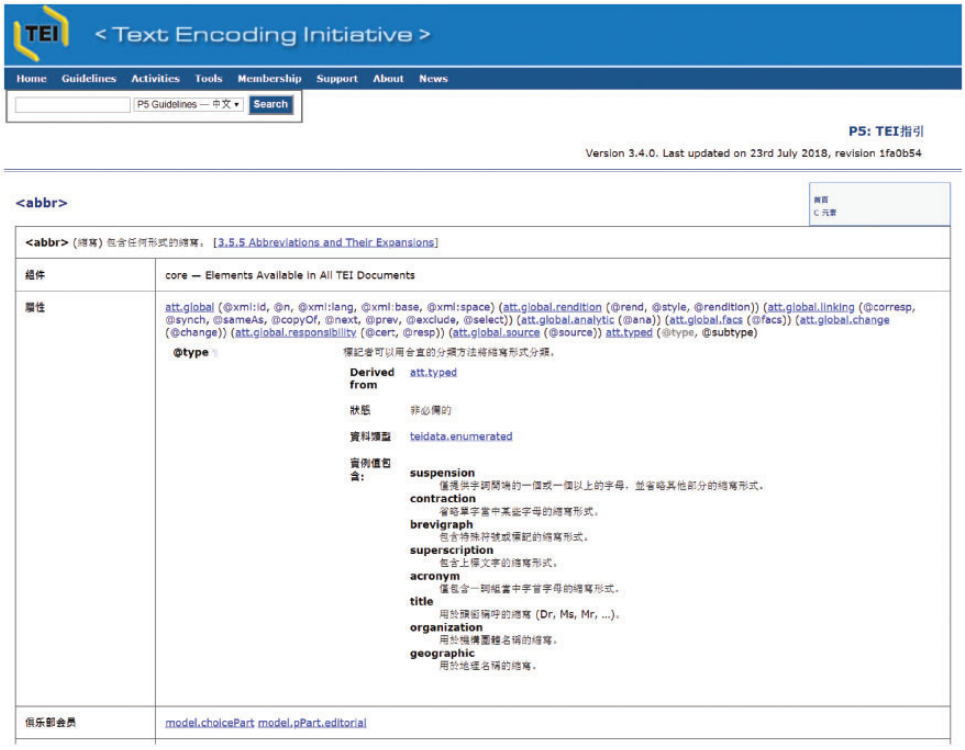

Fig. 13 The TEI Guidelines reference page for the <abbr> element in Chinese[45]

16 Conclusion

16.1 Why do these myths exist?

Some of these myths and misconceptions have grains of truth in them. If it seems as though I have skimmed over the complexities of some problems, this is merely because of the number of myths examined. Although the TEI Guidelines cover a large number of textual phenomena in general ways, there are ways to control the size and specificity of the scheme for any project. While XML does not encode multiple hierarchies when used for embedded markup, there are built-in methods for dealing with these, stand-off markup options, and scripts for alternating between hierarchies. Although there is truth in these myths, it is also possible that there are other reasons for why these misconceptions exist.

When it started as an academic project, the Text Encoding Initiative was not part of the mainstream, but now, given its near ubiquitous acceptance in Digital Humanities for digital textual studies it can truly be considered so. It is not a ragtag group of rebels but has become the establishment which people naturally want to challenge.[40] Another factor leading to misconceptions may be that the TEI Guidelines are often taught in very intensive workshops: this means that people are required to ingest large amounts of information and new concepts in a very short period of time. It is therefore unsurprising that some misunderstandings might occur with such necessarily compressed teaching methods. As part of that teaching, it is sometimes easier to focus on the facts as instantiated by rules and precepts (e.g. 'each TEI document must have a <titleStmt> with a <title> inside') rather than explaining the underlying concepts of why something is being encoded in this manner. Ideally TEI pedagogy would have the leisure to explain both the precepts (the 'how') and the underlying concepts (the

'why'). The best teachers I have known do this as much as possible. Similarly misunderstandings occur because encoders have only read part of the TEI Guidelines (say one or two chapters) and so are unaware of some of the overall infrastructure or possibilities instantiated by other parts of the recommendations. Finally there is the, thankfully rare, wilful misunderstanding for more strategic reasons (such as promoting one's own solutions to the supposed strawmen problems).

16.2 What can we do about these myths?

Combating ignorance of the TEI Guidelines is not an easy task, and clearly more and better teaching materials should improve this. The open training materials already produced by the TEI community go a long way to rectifying this, but perhaps more of these should focus on the why of text encoding in conjunction with the how. Since TEI pedagogy is often compressed into intensive training followed by practical application for a very specific research project, it might be helpful to encourage longer term TEI learning, through creating more self-tuition materials and building teaching into other longer courses. It might also be useful to provide a greater number of shorter, easily digested, introductions to the TEI, accompanied perhaps with surveys of types of encoding and larger full-worked examples demonstrating the whole project arc from encoding to publication and analysis. The TEI By Example project was a good idea and is a reasonable place for those wishing to teach themselves the TEI to start to get basic introductions to a variety of TEI encoding tasks. However, at almost a decade old, and as the TEI continues to develop, TEI By Example becomes increasingly out of date.[41] This is not to say that the TEI Guidelines themselves could not make better use of their own examples, building a corpus of more fully worked detailed encoding samples.[42] Another approach would be the encouragement of new users to discuss more, and more freely, on TEI-L (the TEI community's discussion list), so that misunderstandings can be openly rectified in a spirit of cooperation. However, encouraging that kind of open participation by users new to a domain of knowledge, like the TEI Guidelines, is always difficult. The TEI is a community, with elected volunteers from that community helping to steer it, not a priesthood with special gurus.[43] It is an important aspect of the TEI that anyone can change it, not only in their own local customizations but through participation in the community, submitting feature requests, training others to understand it, or helping to dispel myths and misconceptions in public forums.

References

Bański, P. (2010). Why TEI stand-off annotation doesn't quite work: and why you might want to use it nevertheless. In Proceedings of Balisage: The Markup Conference 2010. https://www.balisage.net/Proceedings//vol5/html/Banski01/BalisageVol5-Banski01.html.

Bauman, S. (2011). Interchange vs. interoperability. In Proceedings of Balisage: The Markup Conference 2011. http://www.balisage.net/Proceedings/vol7/html/Bauman01/BalisageVol7-Bauman01.html.

Burnard, L. and Rahtz, S. P. Q. (2004). RelaxNG with Son of ODD. In Proceedings of Extreme Markup Languages 2004 Conference. http://conferences.idealliance.org/extreme/html/2004/Burnard01/EML2004Burnard01.html.

Burnard, L. (2013). Resolving the Durand Conundrum. *Journal of the Text Encoding Initiative*, **6**. http://jtei.revues.org/842.

Cummings, J. (2008). The text encoding initiative and the study of literature. In Schreibman, S. and Siemens, R. (eds), *A Companion to Digital Literary Studies*. Oxford: Blackwell. http://www.digitalhumanities.org/companionDLS/.

Cummings, J. (2014). The compromises and flexibility of TEI Customisation. In Mills, C., Pidd, M. and Ward, E. (eds), *Proceedings of the Digital Humanities Congress 2012. Studies in the Digital Humanities*. Sheffield: HRI Online Publications. https://www.dhi.ac.uk/openbook/chapter/dhc2012-cummings.

DeRose, S. (2004). Markup overlap: a review and a horse. In Proceedings of the Extreme Markup Languages 2004 Conference. http://xml.coverpages.org/DeRoseEML2004.pdf.

Holmes, M. (2017). Whatever happened to interchange?. *Digital Scholarship in the Humanities*, **32**(Suppl 1): i63–8. https://doi.org/10.1093/llc/fqw048.

Rahtz, S. P. Q. and Burnard, L. (2013). Reviewing the TEI ODD System. In Proceedings of the 2013 ACM Symposium on Document Engineering, DocEng '13. ACM. http://doi.acm.org/10.1145/2494266.2494321.

Renear, A., Mylonas, E., and Durand, D. (1996). Refining our notion of what text really is: The problem of overlapping hierarchies. In Hockey, S. and Ide, N. (eds), *Research in Humanities Computing 4: Selected Papers from the ALLC/ACH Conference*, Christ Church, Oxford, April 1992, pp. 263–80. See also an earlier (1993) version of this at http://cds.library.brown.edu/resources/stg/monographs/ohco.html.

Turska, M., Cummings, J., and Rahtz, S.P.Q. (2016). Challenging the myth of presentation in digital editions. *Journal of the Text Encoding Initiative*.

Unsworth, J. (2011). Computational work with very large text collections. *Journal of the Text Encoding Initiative*.

Notes

1 This article is based on a paper given at the Digital Humanities 2017 conference in Montreal, Canada. The original slides that accompanied the article are available at https://slides.com/jamescummings/tei-myths. From the first release of the TEI P5 Guidelines (28 October 2007), the TEI Consortium moved away from having major releases every few years to a system of rolling version releases every 6 months. At time of writing, the current version of the TEI Guidelines is TEI P5 3.4.0. The experiences of other members of the TEI Technical Council and TEI Community at large over many years have been invaluable to this article.

2 I discuss this frustrating temptation of the 'religious' zealots who often argue that we should latch on to their preferred new technology, dismiss the old, and ignore any costs of doing so in a blog post at: https://faqingperplxd.wordpress.com/2015/05/14/childish-toys/.

3 The number of elements the TEI Guidelines currently include is available on the element reference page from version 3.2.0 onwards, http://www.tei-c.org/Vault/P5/3.4.0/doc/tei-p5-doc/en/html/REF-ELEMENTS.html.

4 TEI Guidelines, Chapter 23: 'Using the TEI', Section 23.3 'Customization' http://www.tei-c.org/Vault/P5/3.4.0/doc/tei-p5-doc/en/html/USE.html#MD.

5 See the latest EpiDoc Guidelines at http://www.stoa.org/epidoc/gl/latest/. The EpiDoc Guidelines are a pure TEI subset which were originally conceived as a markup system for classical epigraphical documents but have since been expanded to encompass other ancient documents with similar textual phenomena and challenges (such as papyri).

6 Document Type Definitions (DTDs) should be considered deprecated in my opinion, since they are no longer fit for purpose with modern XML technologies (e.g. with multi-namespaced documents).

7 There is another approach not mentioned here whereby a TEI ODD customization file may include elements directly from a module while not including the module itself. However, this can be dangerous, since the element will not necessarily benefit from its membership in, for example, attribute classes that are locally defined in the unreferenced module.

8 The TEI ODD XML snippets in the figures here make the assumption that they are nested inside a <schemaSpec> element as part of a well-formed and valid TEI ODD customization.

9 Similar mechanisms, some discussed further below, exist for customizing elements themselves, or indeed the classes and modules of which they are a part.

10 As with all the examples given, this is a real but anonymized example with the name of the desired element changed to protect the naive researcher. I do not feel naming and shaming is in the best interests of the community.

11 Another answer is to extend the TEI through use of customization as described later, but this should not be preferred when there is a satisfactory element that already exists, such as in this case.

12 TEI Guidelines, '<notatedMusic> reference page', http://www.tei-c.org/Vault/P5/3.4.0/doc/tei-p5-doc/en/html/ref-notatedMusic.html.

13 The literate programming methodology is based on the work on Donald Knuth in which human-readable program documentation and code are interspersed and used as a source for both documentation and compiled source code. In the case of TEI customization, the source documentation may be mixed with schema specifications, from which documentation, schema specifications, and even processing model pipelines, may be generated.

14 In both these cases, the TEI ODD customization has @xml:lang and @versionDate attributes on these elements. The first of these specifies the language of the content (to aid the TEI and others in their desires to internationalize) and the second a date at which this was last changed or updated. These attributes are useful for those wanting to track when particular changes were made which is useful when providing translations of content in more than one language.

15 The original class memberships of the <name> element, shown here for completeness, are commented out.
16 Raffaele Viglianti is creating a new customization tool (currently referred to as RomaJS) on behalf of the TEI Technical Council and TEI community. The new tool is undergoing testing and will be provided for open testing in due course from the TEI Consortium website http://www.tei-c.org/.
17 Though work has been ongoing on the creation of an element for describing objects for many years, see https://github.com/TEIC/TEI/issues/327 for more information.
18 See for example Dot Porter's VisColl tool https://github.com/leoba/VisColl which provides a more structured approach to collation.
19 The elected TEI Technical Council considers all requests for changes to the TEI Guidelines submitted as issues to the GitHub repository https://github.com/teic/tei. Having a working customization is not a requirement but can speed up the process.
20 If really arguing for this new element, one would usually provide a number of examples of how it was to be used and why it was needed, as well as countering obvious queries such as why existing mechanisms like using <name type="special"> were not sufficient.
21 This uses a pure TEI content model, in some earlier versions of the TEI these were formulated in RELAX NG. For information on the move away from RELAX NG content models, see Burnard, 2013.
22 Formats currently supported include bibtex, cocoa, csv, docbook, docx, dtd, epub, html(5), xsl-fo, json, InDesign, latex, markdown, mediawiki, nlm, odd, pdf, rdf, relaxng, slides, txt, wordpress, xlsx, xsd, and others. While many of these are legacy conversions created for particular applications with a number of built-in assumptions, they often provide basic conversions or examples for further customization.
23 The OxGarage conversion framework provides a frontend and RESTful Web API to the TEI Consortium's XSLT stylesheets and enables the pipelining of multiple conversions going through a number of different formats. In general, it uses TEI as a pivot format where feasible. It is used by other services like Roma (and the new RomaJS). See http://oxgarage.tei-c.org/.
24 The TEI Technical Council maintains the TEI Guidelines, the transformations necessary to produce various Web versions of them, associated software, build infrastructure, and various GitHub repositories for all these. Additional stylesheet conversions can be provided to the TEI Technical Council for inclusion in their offerings but will be a very low priority for maintenance or support compared with those necessary for TEI Consortium outputs.
25 See http://teiboilerplate.org, http://teic.github.io/CETEIcean/, http://teipublisher.com and http://visualizationtechnology.wordpress.com/ for more information about these open source software tools.
26 See http://www.oxygenxml.com/ for information about this editor. The oxygen-tei package is provided by the TEI Consortium; see https://github.com/TEIC/oxygen-tei.
27 See http://tapasproject.org for more information about TAPAS.
28 I choose to differentiate here between 'stand-off markup' where annotations are stored separately and often used to create new structures that cross hierarchies from a number of existing annotations (for example with the <join/> element) and a simpler 'out-of-line' markup where additional annotation (whether part of a conflicting hierarchy or not) is stored separately and points to a single location in the original.
29 See http://www.tei-c.org/Vault/P5/3.4.0/doc/tei-p5-doc/en/html/SA.html#SAAG for more information on the aggregation of fragmentary structures.
30 More detailed discussion of this approach is available in the TEI Guidelines, Chapter 12 'Critical Apparatus', Section 12.2.4 'Other Linking Methods' http://www.tei-c.org/Vault/P5/3.4.0/doc/tei-p5-doc/en/html/TC.html#TCAPLN.
31 There do exist some generalized tools for working in an out-of-line method, see for example Raffaele Viglianti's Core builder http://raffazizzi.github.io/coreBuilder/.
32 See the chapter of the TEI Guidelines on Non-hierarchical Structures at http://www.tei-c.org/Vault/P5/3.4.0/doc/tei-p5-doc/en/html/NH.html for more consideration of strategies for handling overlap in a TEI context.
33 A simple demonstration of this is visible in https://github.com/TEIC/Stylesheets/blob/dev/tools/processpb.xsl which creates files with a non-TEI <page> elements fragmenting any other hierarchies based on <pb/> elements in the source document.
34 There has been much (digital) ink spilt over solutions to the overlap problem over the years, see also https://www.balisage.net/Proceedings/topics/Concurrent_Markup~Overlap.html.
35 See http://www.tei-c.org/Vault/P5/3.4.0/doc/tei-p5-doc/en/html/USE.html#CF for more information about TEI Conformance. The term is most often abused in research funding proposals.

36 Syd Bauman's definitions (see Bauman, 2011) are useful starting points, and in distinguishing between 'negotiated' and 'blind' interchange he raises the problem of what forms of mediation are necessary and on whose part. Bauman sees interchange as 'negotiated' (requiring human interaction) or 'blind' (only needing documentation). His hope is that we will support 'blind interchange' instead of 'mindless interoperability', since this enables a balance between expressivity and adherence to standards.

37 Martin Holmes' 'CodeSharing API' is also a step in the right direction. It was used to expose the TEI created by the Map of Early Modern London project. See https://github.com/martindholmes/CodeSharing for more information.

38 See for example the manuscript catalogues of the Bodleian Libraries which use a common consolidated TEI ODD customization stored at https://github.com/bodleian/consolidated-tei-schema/. Several catalogues following this one schema are presented separately, e.g. https://medieval.bodleian.ox.ac.uk/.

39 See for example the recent creation of the East Asian TEI Special Interest Group: http://www.tei-c.org/Activities/SIG/EastAsian/.

40 That the TEI Community won ADHO's Antonio Zampolli Award is a particularly fitting sign of this, given Zampolli's involvement as one of the founders of the TEI.

41 This is not to cast any blame on the TEI By Example project, it is just a necessary occurrence for any such endeavour funded as a single project rather than an ongoing service. For more information about TEI By Example, see http://teibyexample.org.

42 I have suggested as much in a paper at the TEI 2018 conference: ' ''Examples work more forcibly on the mind than precepts''—Expanding and improving the use of examples by the TEI Guidelines'.

43 Indeed, the views of the TEI I have presented here are my own and not necessarily representative of the TEI Consortium, and certainly not the TEI community, as a whole.

44 See https://wiki.tei-c.org/ for the TEI Consortium wiki or https://wiki.tei-c.org/index.php/Category:Tools for this 'Tools' category.

45 See http://www.tei-c.org/Vault/P5/3.4.0/doc/tei-p5-doc/zh-TW/html/ref-abbr.html for this page. Note that the values remain as English tokens, but the glosses and some website architecture appear as Chinese.