


RIDE – A review journal for digital editions and resources

published by the IDE

TEI Critical Apparatus Toolbox: Web-based tools for ongoing XML-TEI editions

TEI Critical Apparatus Toolbox, Marjorie Burghart (ed.), 2014-2021. <http://teicat.huma-num.fr/> (Last Accessed: 20.12.2021). Reviewed by  Bastien Dumont (Université de Lorraine), bastien.dumont@univ-lorraine.fr.



Abstract

Despite the flexibility that a TEI-based workflow offers for preparing editions based on collating several manuscripts, textual scholars working with XML-TEI edition files face a lack of domain-specific auxiliary tools. The TEI-CAT partially fills this gap by providing interfaces for visualizing the transcription of different manuscripts and for exploring the encoded variations and detecting common mistakes. While not properly supporting several types of textual variation and editorial intervention, these tools satisfy most of the needs usually arising during transcription and collation. The image annotation tool complements them by allowing users to generate TEI-compliant code. This tool very flexibly describes or comments on facsimile layout and content and thus is particularly helpful in preparing digital editions. Finally, the PDF-via-LaTeX exporting facility extends the scope of the toolbox towards traditional paper-based publishing, which often remains an editor's ultimate goal. However, this tool requires adopting specific encoding practices, which are not made explicit and may not match the user's choices; its output needs manual modifications in some cases. Together, these tools partially illustrate how the XML-TEI format can be used to prepare critical editions. They constitute an important step towards creating a TEI-based digital environment in this field.

General presentation of the TEI-CAT

Intellectual and practical scope

1 Regardless of their purpose and method, scholars preparing critical editions (whether “Lachmannian”, “Bedierist” or focused on the study of textual tradition) often have to register, summarize and account for textual variations across several manuscripts and citations. To date, two approaches have been prevalent. The first one involves entering variations in a collation table, either in a notebook or a spreadsheet, and manually redacting the critical text and the critical apparatuses with a word processor or in a TeX file. The second approach involves using specialized software¹ that provides a unified interface and various auxiliary tools in order to support the editing process from transcribing to exporting the definitive files.

2 Such all-in-one applications have undeniable advantages. However, apart from TUSTEP ([Universität Tübingen 1966–2020](#)), which includes a general-purpose text editor and a full scripting language,² most applications comprise a predefined set of features that the user can hardly extend and that complicate exchanging data with other software — at best, users can export their work for further processing with another software, whose output cannot be reimported and reworked without losing information.³ For instance, Pinche ([2021](#)) relies on the annotation of places, personal names and punctuation marks in the edited text in a detailed and structured way. Other examples involve linguistic tagging or describing the shape of letters and abbreviations. This limitation requires performing such tasks after establishing the critical text, which is not always optimal. A workflow based on a TEI-XML encoded file (as described in [Andrews 2009, 34–74](#)) is highly suited to addressing such needs.

3 When using the [TEI \(Text Encoding Initiative\) encoding standard](#) ([TEI Consortium 2021](#))⁴, which addresses a wide range of needs and provides extension and specialization mechanisms, scholars are virtually unlimited in determining the data structure of their edition file. This is the case because XML-TEI files only contain structured information. As such, they are not meant to be consulted directly by readers but processed by an external programme to produce, for instance, word processor files (to be sent to a publisher), electronic editions (displayed in a web browser), or even work files (e. g. tables containing all variant readings). XSLT ([Kay 2017](#)), a language designed to perform transformations on XML files with the help of stylesheets, is often used to that end (for an example, see [Vaughan 2021](#)). Separating the recording of variants and the

redaction of apparatus entries, which is delegated to further processing (in contrast, notably, to CTE, [Hagel 1997–2021](#)), also ensures a homogeneous syntax in the critical apparatus (see [code example 1](#)) and permits its adaptation to the publisher's requirements by modifying the transformation stylesheet. On the one hand, this requires time and programming skills, but on the other it enables recording a lot of information in a single XML file, only part of which will be used and preserved in every output.

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
<teiHeader>
[... ]
<bibl source="https://www.augustinus.it/latino/cdd/cdd_17.htm"/>
[... ]
</teiHeader>
<text>
[... ]
<body>
<p xml:lang="la">Et quia in prophetia vox eius agnoscitur:
<seg xml:id="cit1">Non <app>
<lem wit="#A" cert="high">derelinques</lem>
<rdg wit="#B #C" type="grammatical">derelinquas</rdg>
</app> animam meam in inferno</seg>,
<seg xml:id="cit2"><app>
<lem wit="#A #C" cert="high">eumdem</lem>
<rdg wit="#B" cause="haplography">eum</rdg>
</app> deduxit ad inferos et reduxit</seg>.</p>
</body>
<standOff type="references">
<spanGrp type="biblicalReferences">
<span target="#cit1">
<ref type="literal" cRef="Ps 15:10"/>
<ref type="allusive" cRef="Ac 2:27"/>
</span>
<span target="#cit2">
<ref type="allusive" cRef="Ac 2:31"/>
</span>
</spanGrp>
</standOff>
</text>
</TEI>
```

Code 1: Edition file with some annotations and the identification of biblical references. Note that the references have been separated from the edited text to avoid overloading.

4 However, while users might appreciate being able to write custom programmes to address specific needs, they benefit from performing common tasks with existing general-purpose tools. Unfortunately, few applications are directed towards critical editing based on XML-TEI and mostly provide interfaces for visualizing achieved editions.⁵ What distinguishes the TEI-CAT is that it provides a set of tools for work in progress. Moreover, contrary to publication software, the two main tools in this toolbox deal almost exclusively with textual variation, which most editors will likely encode in the same way,⁶ and simply ignore more project-specific features. Overall, then, the TEI-CAT truly contributes to developing common tools for common tasks.

Development of the TEI-CAT tools

5 The TEI-CAT (originally called *TEI Critical Edition Toolbox*) was opened for testing in June 2014.⁷ It has been programmed almost entirely by Marjorie Burghart, a medieval Latin philologist, who felt that the existing tools for interacting with XML-TEI editions lacked either sufficient support for the TEI “Critical Apparatus” module or features required to work on ongoing projects (e. g. automatic error detection). To overcome this shortcoming, Burghart decided to make available the scripts that she had written for herself ([Burghart 2019](#)). This initiative has been part of her broader commitment to making XML-TEI based solutions for critical editing more widely known and accessible. As a member of the TEI Consortium’s [Critical Apparatus Workgroup](#), she has also contributed to improving standards. She has also helped create an [online course](#) and a free ebook on encoding critical editions with the TEI ([Burghart, Cummings, and Pierazzo 2017](#)).

6 Originally, the TEI-CAT consisted of two interfaces, which still exist today. Besides conforming to XML and TEI specifications, *Check your encoding* also helps screen apparatus entries for e. g. the absence of a required witness or the mention of two readings for the same witness. *Display parallel versions* exhibits the critical text and the texts of selected witnesses, thus facilitating the verification of collation against every manuscript and the examination of variants. Both interfaces insert anchors in the displayed text that make the individual apparatus entries accessible. What follows refers to these interfaces as the *core tools* of the TEI-CAT, not only because they were released first, but also because they match the needs of all editors using TEI.

7 The TEI-CAT has benefited from scholars and students collaborating on its development during two hackathons and a training week in 2015 and 2016. Some new features have been worked on at that time ([Burghart 2019](#)). Whereas these have never been released, Burghart has published three *additional tools*. (1) A tool for generating TEI <zone> elements from an image was added in 2017, designed by Chris Sparks ([Sparks 2017](#)). Two others followed in 2018: (2) an interface for generating a traditional print edition from an XML-TEI file; (3) a tool for displaying statistics about the XML elements used. In September 2021, [a new version of Annotate an image](#) was released ([Sparks and Vaňková 2017–2019](#)). As this timeline suggests, and as Burghart has confirmed, project changes result mainly from user suggestions, contributions and requests.

Architecture and infrastructure

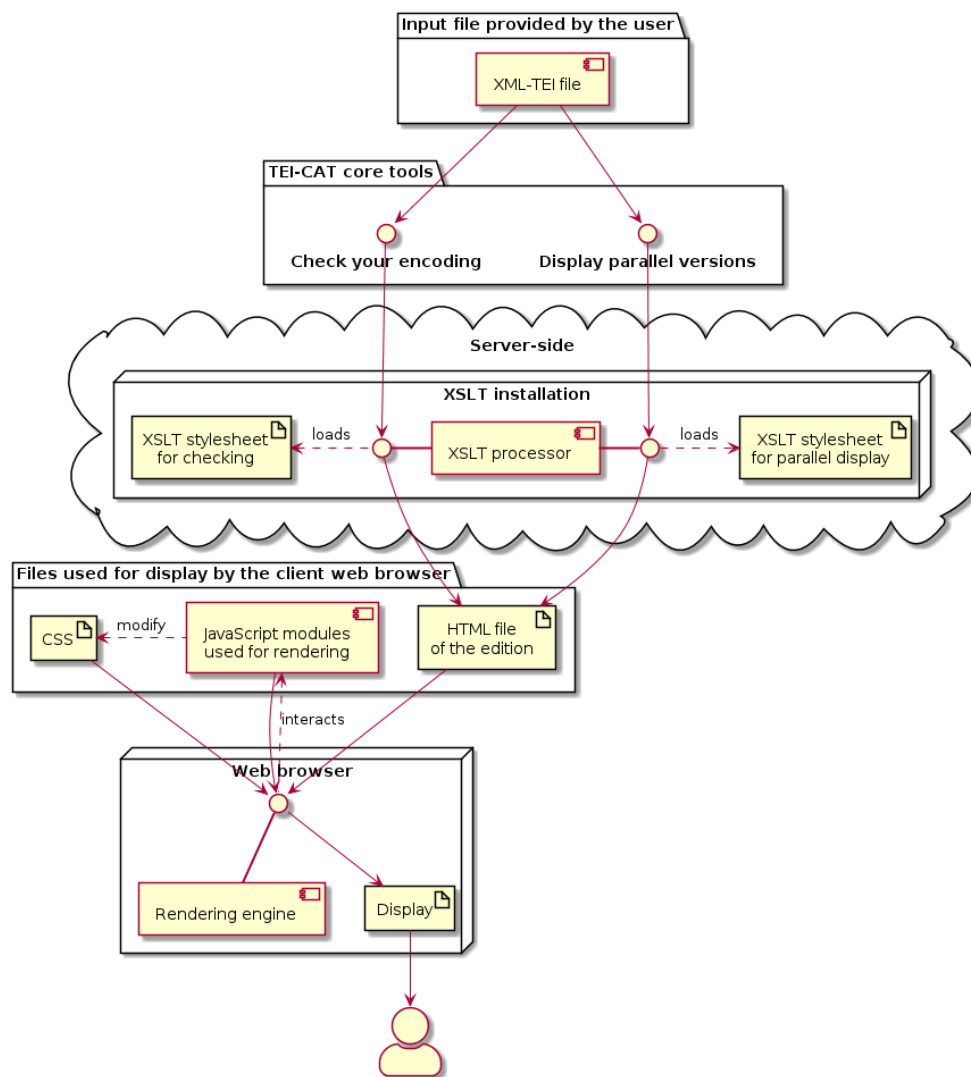


Fig. 1: Processing of an XML-TEI file by the TEI-CAT (core tools)

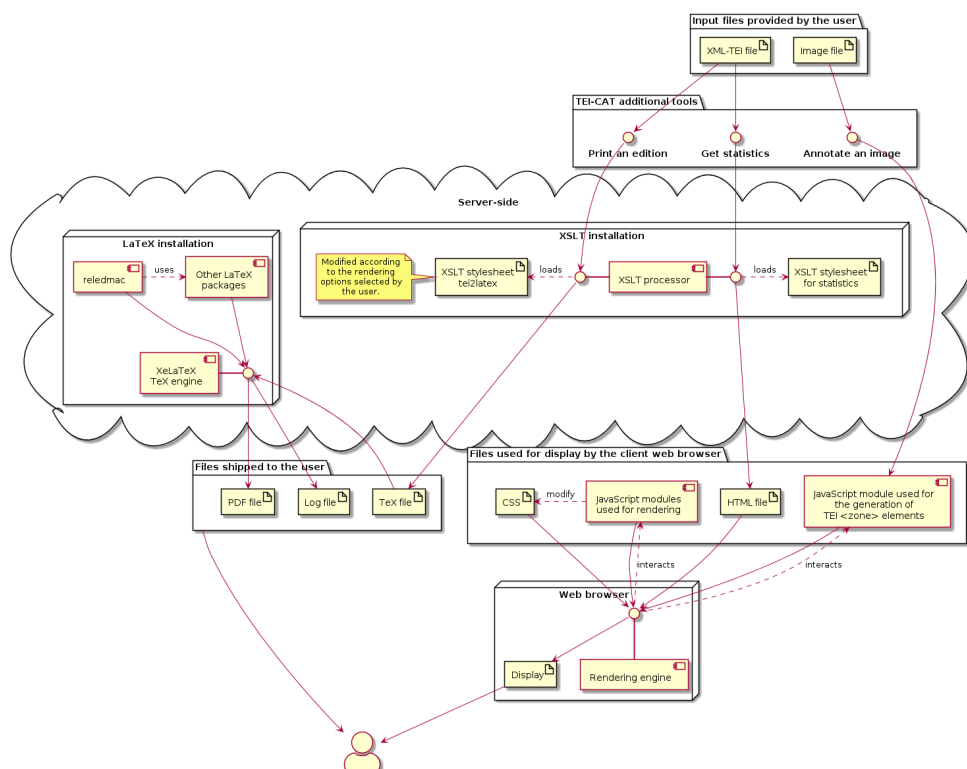


Fig. 2: Processing of an XML-TEI or an image file by the TEI-CAT (additional tools)

8 Generally speaking (see [fig. 1](#) and [fig. 2](#)), the XML-TEI file uploaded by the user is processed by an XSLT processor using the appropriate stylesheets. The generated HTML files are rendered by the browser with the help of the CSS and JavaScript files retrieved from the server. However, two tools depart from this path to some extent. Relying exclusively on a JavaScript program called TEI Zoner, *Annotate an image* does not involve any server-side processing: the programme is executed directly in the user's web browser. As for *Print an edition*, the XSLT processor generates a TeX file, taking into account user options. This TeX file is then processed on the server by a LaTeX engine, which outputs a PDF and a log file. Finally, all three files are bundled into a ZIP archive, which is shipped to the user. The choice of LaTeX is motivated by the fact that, being based on TeX, it produces PDF files of very good typographic quality. It can also be used together with *reledmac*, a package specifically designed for printing critical editions ([Rouquette 2011–2021](#); reviewed in [Dunning 2020](#), who also provides a general introduction to LaTeX).⁸ Finally, it enables the user to manually adjust the automatically generated TeX file and to process it with a local LaTeX installation.

9 Usually, all tools return their output after some seconds. They can even handle rather long and complex editions,⁹ provided that not too many witnesses are selected at

a time. The printing tool may take longer, depending on the time needed by LaTeX to compile the PDF file.

Evaluation

Methodology

10 Given that the core tools share a similar interface and common features (e. g. the generation of the critical text), I review them together. To that end, I have used the test file provided by Marjorie Burghart,¹⁰ which documents part of the behaviour of the core tools. However, being a sample file rather than a comprehensive test input, it does not address some usual textual phenomena (e. g. blank spaces, duplication of words, transposition of whole segments). The same could be said of a handful of frequent encoding features such as the presence of XML elements (e. g. <abbr>) in a <lem> element. To supplement it, I have written a set of additional tests.

11 All the test files, along with a list of the tested features and the detailed results, can be retrieved [from Zenodo](#).¹¹ For each set of tools, I first comment on the interface and then synthesize the outcome of my tests.

Core tools: Checking and visualizing ongoing editions

12 The interface is simple, intuitive and well-described on the [help page](#). The user uploads the XML-TEI file. Then, in *Display parallel versions*, he or she selects the texts to be compared (the critical text and/or one or more witnesses). Texts are displayed side by side (see [fig. 3](#)). Every variation is marked by an anchor, which can be clicked to display the corresponding apparatus entry. Editorial notes (in <note> elements) are rendered as well. In the witness text, passages containing variant readings are highlighted in different colours according to the relationship between the displayed text and the lemma.

13 Hence, *Display parallel versions* addresses the needs of any editor working with several manuscripts. Indeed, it is easier to check transcription accuracy from a continuous text than from the XML-TEI file or even from a collation table. Variant highlighting provides a general idea of the differences between manuscripts before studying them in greater detail and helps study their peculiarities.

14 In *Check your encoding*, only the critical text appears, exactly as in *Display parallel versions*. The user can dynamically select which encoding anomalies or broader

features should be highlighted. Especially appreciated is the ability to highlight those apparatus entries in which some witnesses are missing or recorded twice: these anomalies appear quite frequently, often due to a siglum having been mistyped. Other most common mistakes are tracked too, such as forgetting to define the critical text in a variant location.

- when this witness has the same reading as the lemma, the text is highlighted in white
- when this witness has a different reading from the lemma, the text is highlighted in orange
- when there is no lemma for an apparatus entry, the text of the witness is highlighted in yellow
- when an apparatus entry does not give any intelligible reading for this witness, question marks are highlighted in red ????

Critical text	Text according to A
<p>Liber I CAPUT 1</p> <p>Magnus es, domine, et laudabilis valde: magna⁽¹⁾ virtus tua, et sapientiae tuae non est numerus. et laudare te vult homo, aliqua portio creaturae tuae⁽²⁾, et homo circumferens { mortalitem A B C; fragilitatem D E }⁽³⁾ suam, circumferens testimonium peccati sui et testimonium, quia superbis resistis: et tamen laudare te vult homo, aliqua⁽⁴⁾ portio creaturae tuae. Tu excitas, ut laudare te delectet, quia fecisti nos ad te et inquietum est cor nostrum, donec requiescat in te. Da mihi⁽⁵⁾ mihi A D] nobis B legere, utrum sit prius inv C; om. E scire te prius sit an invocare te. sed quis te invocet nesciens te? Aliud enim pro alio⁽⁶⁾ potest invocare nesciens. An natius</p>	<p>Liber I CAPUT 1</p> <p>Magnus es, domine, et laudabilis valde: magna⁽¹⁾ virtus tua, et sapientiae tuae non est numerus. et laudare te vult homo, aliqua portio creaturae tuae⁽²⁾, et homo circumferens mortalitem⁽³⁾ suam, circumferens testimonium peccati sui et testimonium, quia superbis resistis: et tamen laudare te vult homo, aliquando⁽⁴⁾ portio creaturae tuae. Tu excitas, ut laudare te delectet, quia fecisti nos ad te</p>

Fig. 3: Parallel display with an apparatus entry showing up.

C.[†] Mira refers; sed rumpe moras oculoque[†] sequaci[†] quam primum nobis divinum perlege[†] carmen.

C.[†] 'Qui iuga, qui silvas tueor[†], satus aethere Faunus, haec populis ventura cano: iuvat arbore sacra laeta patefactis[†] incidere carmina fatis[†]. vos o praecipue nemorum gaudete[†] coloni, vos populi gaudete mei: licet omne vegetur securo custode pecus[†] noturna[†] pastor claudere fraxinea[†] nolit[†] praesepia[†] crate,

Fig. 4: Some verses of poetry in the parallel view.

17 As advertised, the core tools effectively support either positive or negative apparatus entries: [fig. 6](#) illustrates what kind of heuristics the stylesheet uses to handle ambiguous cases. However, it would seem more logical to determine the convention followed at the file level rather than for every apparatus entry. In files encoded with a positive apparatus, this would permit highlighting passages where the critical text is not attested in any manuscript and thus results from the editor's reconstruction. However, *Check your encoding* partly offsets this shortcoming by highlighting those variant locations where the readings of selected witnesses are missing.

18 Regarding the recognition of textual features, only two cases documented in the sample file provided by the author are poorly handled. Variants inside a group (<rdgGrp> element) are marked as illegible in the corresponding witness texts. When a given witness is attributed to both the lemma and a variant reading in the same passage, its text in parallel display reproduces that of the lemma instead of indicating an error.

19 Conversely, the core tools fail the majority of the supplementary tests. This comes as no surprise, since these tests mostly concern features not stated as supported by the documentation, namely, the [help page](#) and Burghart's [test file](#). Supported undocumented features include the correct display of right-to-left alphabets, XML elements inside a <lem>, additions, transposed text segments (although the apparatus yields strange results), scribal abbreviations in the lemma or outside a variant location. Unsupported or buggy features include the indication of blank spaces, duplication of expressions or, among possible mistakes, the use of a witness in contexts where this should remain unmentioned (e. g. in a lacuna). However, these shortcomings do not prevent using these tools, since at least the displayed text is generally correct or understandable.

Additional tools: Annotating an image and generating a traditional PDF edition

20 The [basic version](#) of *Annotate an image* lets the user swiftly and easily draw zones on an image and export their coordinates in TEI format. However, it always displays the image in full size, which can be problematic with high-quality reproductions. Nor can the zone be adjusted once drawn. The basic version has been superseded by the [new version](#), which enables scaling images, as well as moving and deleting zones individually. The added features allow categorizing and annotating zones. The help

message is very clear and comprehensive. Although I have not used this tool in real projects, it would seem very helpful for those wishing to include annotations linked to manuscript images in their XML-TEI file (e. g. to publish an electronic edition displaying interactive images alongside the text).

21 At first sight, *Print an edition* looks very attractive. Besides relieving editors of the burden of programming a very complex XSLT stylesheet, the TEI-to-LaTeX transformation provides numerous options for page layout, the formatting of the critical apparatus, the indices, etc. It supports a two-level apparatus containing variant readings and sources. Taken together, the [sample file](#) and the indications in the customization interface provide much information about the supported encoding. Those needing further customization can download and modify the XSLT stylesheet (used to process the XML file). This, of course, requires understanding both XSLT and LaTeX, especially *reledmac*.

22 However, it supports neither variant locations nested in variant readings nor explicitly encoded duplications nor transposed spans of text (i. e. neither *iter.* nor *transp.* can be generated in the critical apparatus: the entire contents of each reading is written out instead). Besides, very long chunks of complicated and sometimes duplicated code make the default XSLT stylesheet difficult to read and modify. Nevertheless, users dissatisfied with the output can use the generated LaTeX file as a basis and modify it by hand. Even those who wish to implement their own stylesheets may benefit from studying Burghart's sheet.

23 Although perhaps beyond the scope of a single-person project, *Print an edition* might serve to more broadly reflect on how to most practically express traditional paper-based editions in form of an XML-TEI model, ideally embodied in a constraining RNG schema. A consensual, well-designed set of conventions with explicit theoretical, albeit traditional foundations, would be suitable for many projects, and would prevent their authors from reinventing the wheel. Such a specification could also be appropriated by other editing and publishing software in the field of textual criticism providing XML-TEI output ([Pierazzo 2019](#); this approach is exemplified by [Camps 2016](#)).

24 Admittedly, programming a more feature-rich and yet reliable set of stylesheets would be a fairly complex and difficult endeavour. However, some design decisions could make this somewhat easier. First, a comprehensive test suite would permit tracking the progress made in supporting the designed model and ensure that no modification introduces bugs. Secondly, the respective responsibilities of the XSLT stylesheet and of

LaTeX macros could be better defined. Currently, for a PDF output similar to [fig. 7](#), XSLT generates the type of code as shown in [example 2](#), which mixes structural commands (e. g. `\edtext` or `\pstart ... \pend`) and formatting instructions (e. g. `\vspace`).

```
\pstart Libero justo laoreet sit amet. In \edtext{pellentesque}
{\lemma{pellentesque} \Afootnote{pellemque \emph{T}, pelles
\emph{P}}}} massa placerat dui.
\pend \vspace{0.1cm}
\pstart Vel elit scelerisque mauris pellentesque pulvinar
pellentesque habitant. Velit egestas dui id ornare arcu odio
ut sem. \edtext{Venenatis urna cursus eget nunc scelerisque
viverra. Eu volutpat odio facilisis mauris sit amet massa
vitae.}{\lemma{} \Bfootnote[nosep]{Augustin, \emph{En. in Ps.},
37, 3 (PL 36, 397; CCSL, 38, 384)}}
\pend \vspace{0.1cm}
```

Code 2: Sample LaTeX code generated by `tei2latex` (slightly modified).¹²

25 This could be made more manageable by using high-level structural commands (see [code example 3](#)):¹³

```
\edparagraph{Libero justo laoreet sit amet. In
\variantLocation{pellentesque}{}{\variant{pellemque}{T}
\variant{pelles}{P}} massa placerat dui.}
\edparagraph{Vel elit scelerisque mauris pellentesque pulvinar
pellentesque habitant. Velit egestas dui id ornare arcu odio
ut sem. \citing{Venenatis urna cursus eget nunc scelerisque
viverra. Eu volutpat odio facilisis mauris sit amet massa
vitae.}{\reference{Augustin, \emph{En. in Ps.}, 37, 3 (PL 36,
397; CCSL, 38, 384)}}}
```

Code 3: Rewriting code as shown in example 2 so that it focuses on structural units.¹⁴

26 These commands could be defined in a separate package file as shown in [example 4](#).

```
\usepackage{reledmac}
\usepackage{etoolbox}
\newcommand{\edparagraph}[1]{\pstart #1
\pend\vspace{0.1cm}}
\newif\ifExistsPrecedingVariation
\newif\ifExistsPrecedingRef
\newcommand{\variantLocation}[3]{%
\ExistsPrecedingVariationfalse
\edtext{#1}{%
\lemma{%
\ifstrequal{#2}{}{#1}{#2}%
}%
\Afootnote{#3}%
}%
}
\newcommand{\variant}[2]{%
\ifExistsPrecedingVariation{, } \fi
#1 \emph{#2}%
\ExistsPrecedingVariationtrue%
}
\newcommand{\citing}[2]{%
\ExistsPrecedingReffalse
\edtext{#1}{%
\lemma{}%
\Bfootnote[nosep]{#2}%
}
```

```

}%
}
\newcommand{\reference}[1]{%
\ifExistsPrecedingRef{, }\fi%
#1%
\ExistsPrecedingReftrue%
}

```

Code 4: Definition of the commands used in Code 3.¹⁵

27 Conceivably, such a LaTeX package could provide some customization options. These could be set in a file generated by the XSLT processor and imported by the main LaTeX file. This would make the XSLT stylesheet more manageable, since it would have to deal only with structure, not with formatting. Later, this would also make it easier to introduce support for *ekdosis* ([Alessi 2020–2021](#)). On the XSLT side, this would only require some additions, to reflect new features, with most of the implementation being done in the package. Needless to say, these are prospective considerations, not actual requirements for TEI-CAT.

Conclusion

28 The TEI-CAT is an important step towards creating a digital ecosystem for critical editing based on XML-TEI. By displaying the recorded text of every witness, *Display parallel versions* enables checking conformity to the manuscript. It also enables the user to visualize apparatus entries. Check your encoding points out possible mistakes or typos in the encoding. Although they do not replace XML editors or the potential use of transcription and collation software, these tools can make transcribing, collating and editing more efficient and more reliable. In sum, they provide the editor with views that help control and interpret data. *Get statistics*, which provides information about XML element use and word count, fits in well with this perspective.

29 *Annotate an image* and *Print an edition* go beyond the production of textual data. The former considerably facilitates linking text and image and as such is oriented more towards producing a digital edition. The latter is modeled on traditional print editions. Users who do not follow the exact same encoding model and feel that some features are missing can modify the stylesheet or rework its LaTeX output. They may thus still gain time compared to writing their own stylesheets from scratch.

30 On the whole, Marjorie Burghart’s work can be characterized as a pioneering general-purpose toolbox that contributes to reducing the need for programming project-specific tools while setting up TEI-based editing workflows. One can only be grateful to

her for making her personal scripts available. Yet, beyond TEI-CAT, significant work remains to be done to create a complete environment. First, in reconstructing the transmission history and the critical text, it would be helpful to be able to explore textual variations in tabular form, with support for sorting, weighting and generating reports. Secondly, unified interfaces for exporting to PDF, DOCX, ODT or HTML would greatly facilitate the publication process. A specialized XML editor, or specific configurations for existing ones, could also make complex source files more manageable.

31 To these ends, some existing components could be built on, such as the [TEI processing model](#) implemented in the [TEI Publisher](#) (2016–2021) or the latter’s annotation editor. However, creating an environment of interoperable software is likely to be hindered, at least for the moment, by the absence of a recognized standard about how to express textual variation using the rich and rather permissive TEI vocabulary. Marjorie Burghart and her TEI-CAT might well play a role in this respect in the future, either directly or as a source of inspiration.

Notes

1. Among others: [TUSTEP](#) ([Universität Tübingen 1966–2020](#)), [Classical Text Editor](#) (CTE, [Hagel 1997–2021](#)) and [ChrysoCollate](#) ([Moureau 2021](#)).

2. See [Christian Griesinger’s review in RIDE 11](#).

3. CTE works around this limitation by letting the user insert arbitrary TEI-XML tags in the text. However, to be exportable to TEI-XML, the apparatus has to be redacted in a machine-readable way, which is generally incompatible with the requirements of a print edition: for instance, following the manual, expressions indicating nested variant locations such as *deus dixit] dominus (christus B) dicit ABC* will not be recognized. The practical consequence is that users must choose between print and TEI output.

4. Especially the modules devoted to the [manuscript description](#), the [representation of primary sources](#) and the [critical apparatus](#).

5. The most prominent ones are [Edition Visualization Technology \(EVT\)](#) and the [Versioning Machine](#) ([Schreibman et al. 2016](#)). [TEI Publisher](#) (2016–2021) has a broader scope.

6. Only parallel segmentation is supported for linking the critical text with the variations. However, the other methods provided by the TEI are less broadly used.

7. <https://web.archive.org/web/20210816160022/https://tei-c.org/2014/06/10/tei-critical-edition-toolbox/>

8. The *ekdosis* package has since been released ([Alessi 2020–2021](#)).

9. Such as [this file](#).

10. Available at <http://teicat.huma-num.fr/pseudo-edition-test-file.xml>. A copy of the version available at the time of writing is included in the ZIP folder mentioned below.

11. I am very grateful to Marjorie Burghart, to whom I sent this folder, for kindly pointing out some errors in my test files. Please note that, as the result of possible work on TEI-CAT, the situation described below may change in the near future.

12. Paragraphs are surrounded by `\pstart` and `\pend`. The critical text in variant locations constitutes the first argument of `\edtext`. The second argument is composed of the lemma, followed by the literal text of the apparatus entry (`\Afootnote`). `\vspace` introduces a gap before the next paragraph. `\Bfootnote` contains an entry in the apparatus fontium.

13. The code examples are merely illustrative.

14. Paragraphs are now set as arguments of `\edparagraph`, whose definition in Code example 4 includes both the `\pstart ... \pend` pair and `\vspace`. The apparatuses are more clearly distinguished by the use of `\variantLocation` and `\citing`. Instead of formatting the critical apparatus directly, the variants are indicated by `\variant`, which takes the variant reading and the list of the corresponding witnesses as arguments. Italics and commas are defined in Code example 4. Note that, in this example, the second argument of `\variantLocation` is empty: Code example 4 shows that it needs to be set only if the lemma in the apparatus differs from the main text.

15. The new commands used in Code example 3 are translated into their *reledmac* equivalents, albeit with some tweaks: for instance, in each critical apparatus entry, a comma and a space are inserted before all variant readings but the first. Using this kind

of definition guarantees that all apparatus entries are formatted according to the same rules, which can then be modified uniformly.

References

- Alessi, Robert. 2020–2021. *ekdosis – Typesetting TEI-XML compliant Critical Editions*. LaTeX.
<https://web.archive.org/web/20221111115050/https://www.ctan.org/pkg/ekdosis>.
- Andrews, Tara L. 2009. “Prolegomena to a Critical Edition of the Chronicle of Matthew of Edessa, with a Discussion of Computer-Aided Methods Used to Edit the Text.” PhD Thesis, Oxford: Oxford University.
<https://web.archive.org/web/20221111115125/https://ora.ox.ac.uk/objects/uuid:67ea947c-e3fc-4363-a289-c345e61eb2eb>.
- Burghart, Marjorie. 2019. “The TEI Critical Apparatus Toolbox: Empowering Textual Scholars Through Display, Control, and Comparison Features.” *Journal of the Text Encoding Initiative* 10.
<https://doi.org/10.4000/jtei.1520>.
- Burghart, Marjorie, James Cummings, and Elena Pierazzo. 2017. *Creating a Digital Scholarly Edition with the Text Encoding Initiative*. Edited by Marjorie Burghart. DEMM.
https://web.archive.org/web/20210816161355/https://www.digitalmanuscripts.eu/?page_id=648.
- Camps, Jean-Baptiste. 2016. “La Chanson d’Otinél : édition complète du corpus manuscrit et prolégomènes à l’édition critique.” Thèse de doctorat, Paris: Université Paris-Sorbonne (Paris IV).
<https://web.archive.org/web/20221111115306/https://halshs.archives-ouvertes.fr/tel-01664932>.
- Dumont, Bastien. 2021. *Tests for the TEI-CAT* (version 1.1).
<https://doi.org/10.5281/zenodo.5270343>.
- Dunning, Andrew. 2020. “Reledmac. Typesetting Technology-Independent Critical Editions with LaTeX.” *RIDE* 11.
<https://doi.org/10.18716/ride.a.11.1>.

- Hagel, Stephan. 1997–2021. *Classical Text Editor* (version 10.4). Windows. Austrian Academy of Sciences, Corpus Scriptorum Ecclesiasticorum Latinorum.
<https://web.archive.org/web/20221104100713/https://cte.oeaw.ac.at/>.
- Eberhard Karls Universität Tübingen. 1966–2020. *Tübinger System von Textverarbeitungs-Programmen*. Linux, Mac OS, Raspbian, Windows. Universität Tübingen, Zentrum für Datenverarbeitung.
<https://web.archive.org/web/20221008175234/http://www.tustep.uni-tuebingen.de/>.
- Kay, Michael, ed. 2017. “XSL Transformations (XSLT) Version 3.0.” W3C. June 8, 2017.
<https://www.w3.org/TR/xslt-30/>.
- Moureau, Sébastien. 2021. *ChrysoCollate* (version 1.0). Browser-based. Université catholique de Louvain.
<https://web.archive.org/web/20221111120045/https://cental.uclouvain.be/chrysocollate/>.
- Pierazzo, Elena. 2019. “What Future for Digital Scholarly Editions? From Haute Couture to Prêt-à-Porter.” *International Journal of Digital Humanities* 1: 209–20.
<https://doi.org/10.1007/s42803-019-00019-3>.
- Pinche, Ariane. 2021. “Édition nativement numérique du recueil hagiographique *Li Seint Confessor* de Wauchier de Denain d’après le manuscrit fr. 412 de la Bibliothèque nationale de France.” Thèse de doctorat, Lyon: Université Jean Moulin (Lyon 3).
<https://web.archive.org/web/20221111120146/https://www.theses.fr/2021LYSE3019>.
- Rosselli Del Turco, Roberto, ed. 2016–2021. *Edition Visualization Technology* (version Beta2). Node.js. EVT Team.
<https://web.archive.org/web/20221013152203/http://evt.labcd.unipi.it/>.
- Rouquette, Maïeul. 2011–2021. *reledmac – Typeset scholarly editions*. LaTeX.
<https://web.archive.org/web/20221111120343/https://www.ctan.org/pkg/reledmac>.
- Schreibman, Susan, Karolina Badzmierowska, and Roman Bleier. 2016. *Versioning Machine* (version 5.0). Browser-based.
<https://web.archive.org/web/20221111120450/http://v-machine.org/>.
- Sparks, Chris. 2017. *TEI Zoner*. Browser-based.

<https://github.com/sparkyc84/tei-zoner>.

Sparks, Chris, and Marie Vaňková. 2017–2019. *tei-zoner-angular*. Browser-based.

<https://github.com/Elfina1039/tei-zoner-angular>.

TEI Consortium. 2021. “TEI P5: Guidelines for Electronic Text Encoding and Interchange.” Text Encoding Initiative. 2021.

<https://tei-c.org/release/doc/tei-p5-doc/en/html/index.html>.

TEI Publisher: The Instant Publishing Toolbox (version 7.1). 2016–2021. Java. e-editiones.org.


<https://web.archive.org/web/20221110134226/https://teipublisher.com/index.html>.

Vaughan, Nicolás. 2021. “TEI-XML to LaTeX Workflow: Issues and Lessons.” *TUGboat* 42 (2): 174–79.

<https://doi.org/10.47397/tb/42-2/tb131vaughan-tei>.

Factsheet

Resource reviewed	
Title	TEI Critical Apparatus Toolbox
Editors	Marjorie Burghart
URI	http://teicat.huma-num.fr/
Publication Date	2014-2021
Date of last access	20.12.2021

Reviewer	
Name	 Dumont, Bastien
Affiliation	Université de Lorraine
Place	Nancy, France
Email	bastien.dumont (at) univ-lorraine.fr

General information		
Software type	What type of software is it? (cf. Catalogue 0.1.1)	Software tool
Identification of the environment	On which platform runs the tool? (cf. Catalogue 1.4)	Web browser
Purpose	For what purpose was the tool developed? (cf. Catalogue 1.5)	developed to accomplish a general task
Funding	Which is the financial model of the tool? (cf. Catalogue 1.6)	Free/open
Maturity	What is the development stage of the tool? (cf. Catalogue 1.5)	Release
Methods and implementation		
Programming Language	Which programming languages and technologies are used? (cf. Catalogue 2.3)	XSLT, Other: LaTeX
Reuse	Does the tool reuse portions of other existing software? (cf. Catalogue 2.3)	yes

Input format	Which input formats are supported? (cf. Catalogue 2.4)	.xml/tei
Output format	Which output formats are supported? (cf. Catalogue 2.4)	.pdf, .html, Other: .tex
Encoding	Which character encoding formats are supported? (cf. Catalogue 2.4)	utf-8
Encoding preprocessing	Is a pre-processing conversion included?	no
Dependencies	Does the documentation list dependencies on other software, libraries or hardware? (cf. Catalogue 3.2)	yes
Dependencies installation	If yes, is the software handling the installation of dependencies during the general installation process (you don't have to install them manually before the installation)?	yes
Documentation and support		
Documentation	Is documentation and/or a manual available? (tool website, wiki, blog, documentation, or tutorial) (cf. Catalogue 3.4)	yes
Documentation format	Which format has the documentation? (cf. Catalogue 3.3)	.html
Documentation parts	Which of the following sections does the documentation contain? (cf. Catalogue 3.3)	Step-by-step instructions, Examples
Documentation language	In what languages is the documentation available? (cf. Catalogue 3.3)	English
Support	Is there a method to get active support from the developer(s) or from the community? (cf. Catalogue 3.4)	no
Form of support	Which form of support is offered? (cf. Catalogue 3.4)	Not applicable (if no support is given)
Issue tracker	Is it possible to post bugs or issue using issue tracker mechanisms? (cf. Catalogue 3.4)	no
Usability and sustainability		
Build and install	Grade how straightforward it is to build or install the tool on a supported platform: (cf. Catalogue 3.6)	straightforward

Tests	Is there a test suite, covering the core functionality in order to check that the tool has been correctly built or installed? (cf. Catalogue 3.7)	no
Portability and interoperability	On which platforms can the tool/software be deployed? (cf. Catalogue 3.8)	Not applicable (if web-based for example)
Devices	On which devices can the tool/software be deployed? (cf. Catalogue 3.8)	Not applicable (if web-based for example)
Browsers	If the tool is web-based: On which browsers can the tool/software be deployed? (cf. Catalogue 3.8)	Unknown
Plugins	If the tool is web-based: Does the tool rely on browser plugins? (cf. Catalogue 3.8)	no
API	Is there an API for the tool? (cf. Catalogue 3.8)	no
Code	Is the source code open? (cf. Catalogue 3.9)	no
License	Under what license is the tool released? (cf. Catalogue 3.9)	No explicit license / all rights reserved
Credits	Does the software make adequate acknowledgement and credit to the project contributors? (cf. Catalogue 3.9)	yes
Registered	Is the tool/software registered in a software repository? (cf. Catalogue 3.9)	no
Possible contribution	If yes, can you contribute to the software development via the repository/development platform?	not applicable
Analysability, extensibility, reusability of the code		
Analysability	Can the code be analyzed easily (is it structured, commented, following standards)? (cf. Catalogue 3.10)	not applicable
Extensibility	Can the code be extended easily (because there are contribution mechanisms, attribution for changes and backward compatibility)? (cf. Catalogue 3.10)	not applicable

Reusability	Can the code be reused easily in other contexts (because there are appropriate interfaces and/or a modular architecture)? (cf. Catalogue 3.10)	not applicable
Security and privacy	Does the software provide sufficient information about the treatment of the data entered by the users? (cf. Catalogue 3.11)	no
Supportability and maintenance	Is there information available whether the tool will be supported currently and in the future? (cf. Catalogue 3.12)	no
Citability	Does the tool supply citation guidelines (e.g. using the Citation File Format)? (cf. Catalogue 3.13)	no
User interaction, GUI and visualization		
User profile	What kind of users are expected? (cf. Catalogue 4.1)	Humanities researcher, Digital humanist
User interaction	What kind of user interactions are expected? (cf. Catalogue 4.1)	Reading, Image editing, Visualization, Comparing
User Interface	What kind of interface does the tool provide? (cf. Catalogue 4.2 and 0.1.1)	Graphical User Interface (GUI)
Visualization	Does the tool provide a particular visualizations (in terms of analysis) of the input and/or the output data? (cf. Catalogue 4.3)	yes
User empowerment	Is the user allowed to customize the functioning of the tool and the output configuration? (cf. Catalogue 4.4)	yes
Accessibility	Does the tool provide particular features for improving accessibility, allowing „people with the widest range of characteristics and capabilities" to use it? (cf. Catalogue 4.5)	no
Personnel		
Editors	Burghart, Marjorie	
Programmers	Burghart, Marjorie	
Designers	Burghart, Marjorie Mercier, Karyn	
Contributors	Sparks, Chris Vaňková, Marie	