Schriften des Instituts für Dokumentologie und Editorik — Band 16

Digitale Edition in Österreich
Digital Scholarly Edition in Austria

---

herausgegeben von | edited by

Roman Bleier, Helmut W. Klug

2023

BoD, Norderstedt

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de/ abrufbar.

Digitale Parallelfassung der gedruckten Publikation zur Archivierung im Kölner Universitäts-Publikations-Server (KUPS). Stand 29. April 2023.

2023

Herstellung und Verlag: Books on Demand GmbH, Norderstedt
ISBN: 978-3-743-102-842
Einbandgestaltung: Stefan Dumont; Coverbild: wurde von Roman Bleier und Helmut Klug für ein KONDE-Poster (DHa 2017) erstellt
Satz: Roman Bleier und LuaTEX

Experience with a Workflow using MS Word and a DOCX to TEI Converter

Markus Ender, Joseph Wang

Abstract

In this contribution we summarize our experience in edition projects with the conversion from DOCX files to TEI-XML files. A tool called DOCX2TEI, developed in our team, is capable of transforming DOCX files automatically to TEI files without the need of manual post-processing of conversion results by editors. However, for this to function properly pertinent programming knowledge is required to make the necessary adaptations on DOCX2TEI. We compare DOCX2TEI with other tools capable of producing TEI files out of DOCX files. At the end of the paper we will present the lesson learned from deploying and using this tool.

Zusammenfassung

In diesem Beitrag schildern wir unsere Erfahrungen in Editionsprojekten mit der Konvertierung von DOCX-Dateien zu TEI-XML. DOCX2TEI ist ein Tool, das im Rahmen dieser Projekte entwickelt worden ist; es verwandelt DOCX-Dateien automatisch in TEI-Dateien, ohne dass die Resultate nachbearbeitet werden müssen. Damit dieser Prozess richtig funktioniert, muss eine Adaptierung des Tools vorgenommen werden, die einschlägige Programmier-Kenntnisse verlangt. Wir stellen DOCX2TEI anderen Tools gegenüber, die ebenfalls dazu verwendet werden können, um TEI-Dateien aus DOCX-Dateien zu erzeugen. Am Schluss des Beitrags geben wir unsere Lehre aus unseren Erfahrungen mit diesem Workflow wieder.

Since the introduction of TEI P5 in 2007 (TEI-Consortium 2019, ch. i.) creating TEI-XML data has become a common task in humanities research. Today most editing projects make use of TEI-XML for the modelling and annotation of primary sources. However, for those scholars not familiar with the praxis of coding, creating TEI-XML can be a frightening task. There are many different reasons for individual scholars to be intimidated by the TEI guidelines. One of them most certainly is that the learning curve for TEI is steep. Since XML is the underlying technology of TEI, a common TEI course usually starts with a basic XML introduction where participants soon get

Digitale Edition in Österreich. Hrsg. Roman Bleier und Helmut W. Klug. Schriften des Instituts für Dokumentologie und Editorik 16. Books on Demand, 2023, 55–68.

the feeling that concepts like *element, attribute, node* and *comment* are confusing and maybe even a little bit frightening. The course will then focus on explaining the different parts of the TEI guidelines. In the end participants will realize that—even though it is not necessary to memorize all 576 elements of the TEI guidelines (TEI-Consortium 2019, Appendix C.1)—coding in XML is still a very different task from using a common word processor. Another difficulty that arises when people start to use TEI is the paradox of choices. While we do welcome the flexibility of the TEI guidelines (the ability of using different approaches to solve one problem is a key feature of TEI), being able to choose is sometimes an excessive demand, especially for beginners. Instead of a single possibility to carry out one task, e.g., how double underlined text passages should be encoded, there are always different solutions to a single problem. These and many more reasons keep humanities scholars from using XML editors.

In this paper we want to share our experience with using an application to convert MS *Word* DOCX files to TEI (DOCX2TEI).[1] There are benefits and challenges using such a converter and we hope that by publishing our experience readers can benefit from our know-how.

1 History of using a DOCX to TEI converter in the research institute Brenner-Archives

In the Research Institute Brenner-Archives (Forschungsinstitut Brenner-Archiv, FIBA)[2] of the University of Innsbruck, TEI had a slow start. While members of FIBA have always embraced digital technology for their editing activities, TEI was first introduced through the cooperation of FIBA and the Wittgenstein Archives in Bergen (WAB) on the project *Culture and Value Revisited* (FWF P19022, led by Allan Janik). Vemund Olstad, then a member of WAB, gave the first formal introduction to XML and the TEI in 2006. However, aside from this cooperation no other projects have adopted TEI.

During the project, project staff also acquired the technical skills for XML processing. These skills were essential when it was decided that the results of the projects *Critical Online-Edition of the Works of Otto Weininger* (FWF P17975), *Critical On-Line-Edition of Otto Weininger's Correspondence* (FWF P20301) and *A Documentary Volume on Otto Weininger and his Works* (FWF P22168, all lead by Allan Janik) should be published as online-accessible digital editions in 2008.

---

[1] As soon as Ficker's correspondence is published online, the entire source code of DOCX2TEI will be publicly accessible. We will also license it under a free license, so other projects may re-use DOCX2TEI or any parts of it for their own purpose.

[2] https://www.uibk.ac.at/brenner-archiv/.

At that time, Intelex, the publisher who wanted to publish the edited volumes of the *Weininger* projects, could only process data in FolioView format or in TEI-XML. The documents of the *Weininger* projects, however, were available only as MS *Word* files. Furthermore, the editors have employed some functions of MS *Word* in order to "mimic" the behavior of web pages, such as adding comments and hyperlinks. The commenting function was used to add additional information to different lemmata and the hyperlinking function was used to link names to biographies and lists of works in arts and literature. Already at that time, MS *Word* (Version MS *Word* 2007) was able to export a *Word* document to XML. The XML used there is a predecessor of the OOXML specification (Kelly 2003). The best technical approach seemed therefore to write an XSLT stylesheet to convert the XML produced by MS *Word* to TEI-XML.

Unfortunately, the conversion from *Word* XML to TEI succeeded only in parts. We had to manually correct several passages of the conversion. One major issue was that the authors have used a special font-face for Greek letters, e.g. the letter 'a' in this font-face encodes for 'α'; and the font-face also used zero space diacritics to form composite accented characters. Therefore, we had to manually check and emend the conversion errors. After the emendations and two years of work, the results of the *Weininger* projects were published using Intelex's publishing platform in 2010 (Hirsch 2011).

After the project, the FIBA technical staff soon realized the potential inherent in the conversion of a text processor format to TEI-XML. Therefore, they started to generalize the conversion workflow so that it could also be used for other scenarios. At the same time, MS *Word* moved their standard format from DOC to DOCX (ECMA-International 2016). In doing so they have also opened up the possibility of transforming *Word* files automatically to TEI. The DOCX format is basically a set of XML files (together with other media files) that are compressed into a ZIP file. Thus, when the project *Ludwig von Ficker as a Cultural Transmitter* (FWF P24283, led by Eberhard Sauermann) and later its follow-up project (*Ludwig von Ficker: Critical Online Edition of the Correspondence and Monograph*, FWF P29070, under the lead of Ulrike Tanzer)[3] were granted, we developed routines to convert DOCX files to TEI-XML and then send the TEI data to an XML database. There were two main reasons for using these conversion routines. The first reason was that about half of the correspondences were already available as transcriptions in *Word* files.[4] Instead of manually copying and pasting the vast amount of data into XML files, it was, of course, easier to convert the data automatically. The second reason was that there simply are too many letters to be manually encoded. Judging from the amount of correspondence there is not enough

[3] One of the main aims of this project was to make Ficker's correspondence (about 17,300 letters) available to scholars and the wider public.

[4] Aside from four published volumes (Zangerle 1986–1996), Ficker's correspondences were constantly being transcribed at FIBA.

time nor personnel resources to encode every aspect of a letter. Therefore, we had to strictly confine our transcriptions and commentaries, and so MS *Word* (even with its limited features) worked fine for the project.

It is worth noting that the nature of the conversion has changed since we moved from the *Weininger* projects to the *Ficker* projects. In the *Weininger* projects, the editing process had already been finished – essentially, the edition texts were not changed anymore. The conversion from DOCX to TEI-XML was therefore a one-time process and afterwards all corrections to the edition – when there had been any – were manually recoded. Furthermore, we were only interested in converting a few *Word* features like character and paragraph formatting and hyperlinking. In contrast to this, the aim of the DOCX to TEI-XML conversion for the *Ficker* projects have substantially changed. The transcriptions of the letters appear in different qualities and stages throughout the edition. Some letters had already been edited and enriched with comments; some were only transcribed and collated; some were just transcribed without the necessary step of collation. The biggest bulk of the letters, however, was not even transcribed. Thus, our expectation was that the majority of files would be changed constantly. Furthermore, at the beginning of the project the direction and goals of the edition (besides providing a reading text) were not yet clearly defined. With these considerations in mind, the conversion process had to be more flexible and it needed to be easily adjustable in case the editing guidelines were revised.

2 The current atatus of the *Ficker* projects and the reuse of the routines for other projects

The main aim of the routine we call DOCX2TEI is to extract data from files in DOCX format written in a specific way.[5] Each file contains at least one letter, but potentially as many letters as possible between Ficker and one of his correspondence partners. Each letter starts with a normalized heading containing the metadata of the letter. Paragraphs concerning provenance and archival notes follow the heading. Then the file contains the letter proper. The letter itself consists of a transcription and annotations on the transcription. The annotations can be either references to places, people, and subjects; or they contain editorial remarks to a lemma. The conversion should also retain paragraphs and character formatting. Each letter should result in one TEI-XML file. Since we will have more than 17,300 TEI-XML files at the end of

[5] There are other features (or requirements) that are not part of the conversion, although they are part of TEI data creation. For example, the linking of references to an authority file is done through a separated mapping tool that maps the unique keys to, e.g., *Gemeinsame Normdatei* (GND, Deutsche Nationalbibliothek 2019). Adding GND URIs to TEI data is done only when full TEI-XML data is required, e.g. for users who want to download the data for themselves, but not during the conversion of DOCX to TEI.

the projects, we have agreed that any correction of the data should happen in the DOCX files. The conversion should reach a quality that manual changes to the TEI document after conversion are not necessary anymore.

The basic idea for the conversion is quite simple: We are using XSL-Transformation for the conversion process. A small number of XSLT stylesheets is used in this process and each of those transforms a single task of the conversion routine. The conversion process starts at the main entry point of the DOCX file. Then, a series of XSL transformations is initiated, each using the results of the previous XSL transformation as input. This routine enables us to debug and improve the transformation step by step.[6] The result of the last transformation will then be sent through an enterprise integration solution to a backend where the data is stored in a database and further data processing that is not considered part of the conversion (like adding certain metadata and authority controlled data) takes place. There is also a frontend that uses the Web-API provided by the backend application. The frontend, however, does not enable scholars to change the TEI data. The frontend is meant for the purpose of monitoring the conversion and doing some data processing, e.g., searching or aggregating data. The frontend provides users with a table of content of all TEI files, a HTML and a PDF representation of every TEI-XML file, a full-text search, and a registry of annotated TEI data.[7]

A few examples should illustrate how a set of stylesheets is organized.

1. One stylesheet is specialized in separating letters from the whole set of correspondence. This stylesheet is capable of creating different TEI files from a single input document/file. This is actually a special case where we have not only one, but many TEI-XML files as a result. Each resulting TEI-XML file is passed on to the following stylesheets individually.

2. A second stylesheet is used to create the TEI header element by parsing the heading and all metadata contained in the first lines. Since the heading (title) of each letter is standardized, we can write a parser to extract information from

---

[6] It should be noted here that changing the order of stylesheets will change the outcome. It could happen that after changing, e.g., stylesheet no. 2, stylesheet no. 4 will not work anymore. Because of this, stylesheets that change the general data structure should actually be applied later in the stylesheet series.

[7] Since we are only concerned with the conversion process in this paper, we will just briefly mention the post-processing and the hardware configuration: the enterprise integration solution is based on *Apache Camel* (https://camel.apache.org/). We have created a backend application based upon *Apache Sling* (https://sling.apache.org/) which also serves as the data storage for the XML data. *Camel* is able to periodically check directories for changed files and process them. The results are then sent to *Sling*. Additionally, we created our own frontend using *AngularJS* (https://angularjs.org/). At the time of writing, we are migrating the backend from *Apache Sling* to a standalone java application using *eXist* (http://exist-db.org/exist/apps/homepage/index.html) as data storage. The frontend will be based upon the *Angular Framework* (https://angular.io/).

the headings. Some effort was put into the parsing of dates, to make sure that uncertain dating is encoded correctly.

3. Several stylesheets are responsible for converting comments to references of people and places. To encode a certain reference, we need to encode both the referring words and the referred entity. In the TEI guidelines, these references are to be written as `<rs>`, `<persName>`, `<placeName>` or `<date>` using the lemma as the text of the XML elements, and the referenced entity is provided as value of an attribute (either `@ref` or `@key`) of the XML element. If we use foot- or endnotes to annotate references in DOCX files we lose the starting point of the annotation. MS *Word* allows users to add comments to a document where the comments consist of a starting XML-element, an ending XML-element and one element for the note itself.[8] We have created a stylesheet for the conversion of these elements to a single XML element. Other stylesheets are responsible for converting these XML elements further to the correct TEI elements. We introduced a special syntax for encoding references, e.g., a comment starting with "P:" denotes a reference to a person and a comment starting with "O:" denotes a reference to a place. What follows "P:" or "O:" is the normalized name of the person respectively the place (see Figures 1 and 2). The normalized names of people and places also act as the `@key` value of the referencing element. If editors need to add additional information on the lemma, they get the possibility of adding this information in a new paragraph within the comment. The information in the paragraph will be converted automatically to a `<note>`-element.

Special care is needed when converting the delimiting milestones (and the nodes between them) into an element. At first, we have simply replaced the milestones with a starting and an ending tag. However, in some cases (e.g. when the commented lemmata are overlapping) we have produced overlapping elements and the result is not a well-formed XML anymore (Wang 2016).

As one can see, by organizing the stylesheets in this way, we can add, switch or omit stylesheets in order to adapt the conversion process for other projects. For example, if in a project each DOCX should also be converted to a single `<TEI>` we can omit the separation stylesheet. Similarly, we can adapt the conversion of comments by adding new rules of conversion. If, e.g., in a project people need to annotate literary works in a certain way, we can easily introduce a new syntax for this task (e.g., "Bibl: the name of author: the title of the work").

Since the beginning, we have also improved several other features of our approach. For instance, the way we extracted metadata from each letter using a parser soon

[8] Note that the starting and the ending points of a MS comment are marked up using XML elements (and not a start- and end-tag).

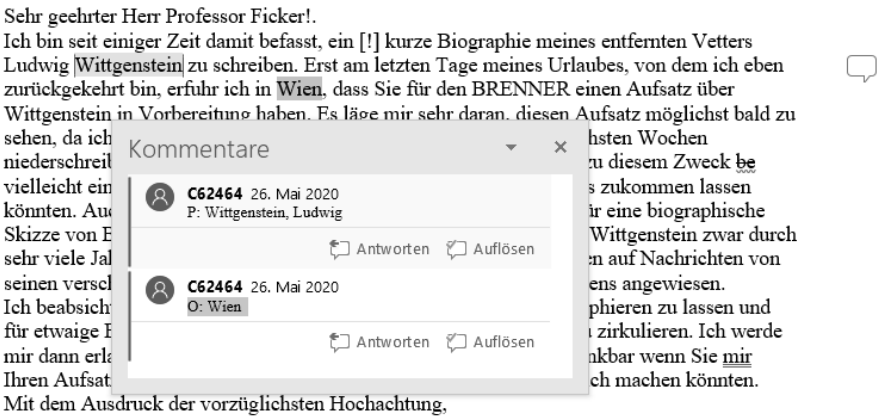

Figure 1: *Word* document with annotations.

```
<p>Sehr geehrter Herr Professor Ficker!.<lb/>Ich bin seit einiger Zeit damit
    befasst, ein [!] kurze Biographie meines entfernten Vetters Ludwig
    <persName key="Wittgenstein, Ludwig">Wittgenstein</persName>zu schreiben. Erst am
    letzten Tage meines Urlaubes, von dem ich eben zurückgekehrt bin, erfuhr ich in
        <placeName key="Wien">Wien</placeName>, dass Sie für den BRENNER einen
    Aufsatz über Wittgenstein in Vorbereitung haben. Es läge mir sehr daran, diesen
    Aufsatz möglichst bald zu sehen, da ich meine Arbeit, wenigstens im ersten
    Entwurf, in den nächsten Wochen niederschreiben möchte. Ich wäre Ihnen sehr
    dankbar, wenn Sie mir zu diesem Zweck <seg rend="strikethrough">be</seg>
    vielleicht einen Bürstenabzug oder einen Durchschlag Ihres Aufsatzes zukommen
    lassen könnten. Auch für alle anderen Mitteilungen über Wittgenstein, die für
    eine biographische Skizze von Bedeutung sein könnten, wäre ich sehr dankbar. Ich
    habe Wittgenstein zwar durch sehr viele Jahre aber nie wirklich gut gekannt, und
    bin im Allgemeinen auf Nachrichten von seinen verschiedenen Freunden aus den
    diversen Perioden seines Lebens angewiesen.<lb/>Ich beabsichtige, den ersten
    Entwurf der Skizze zunächst <seg rend="strikethrough">zu</seg> hektographieren
    zu lassen und für etwaige Berichtigungen und Ergänzungen unter W.'s Freunden zu
    zirkulieren. Ich werde mir dann erlauben, Ihnen ein Exemplar zuzusenden. Ich
    wäre sehr dankbar wenn Sie mir Ihren Aufsatz und etwaige sonstige Mitteilungen
    noch [xxx] zugänglich machen könnten.<lb/>Mit dem Ausdruck der vorzüglichsten
    Hochachtung,<lb/>Ihr sehr ergebener<lb/>FAvHayek<lb/>(Prof. Dr. F.A. von
    Hayek)</p>
```

Figure 2: View of the TEI data in *Oxygen XML Editor*.

seemed to be quite laborious. Instead of parsing the header (a text string) we should have created tabular data at the beginning of each letter containing metadata like "date", "sender", "recipients", and "archival note". The editors could have then entered the data directly into the table. By doing so, the stylesheet for creating `<teiHeader>` would be much easier to write and maintain. Unfortunately, at the time we realized

this the transcription process in the Ficker project was already far too advanced to change the transcription workflow. But other projects have already benefited from this insight!

The correctness of the DOCX to TEI conversion needs to be checked by scholars after the conversion process. On the one hand, the conversion routine may contain bugs and produce incorrect artifacts, and on the other hand, the DOCX file itself may contain errors (e.g. typos and transcription mistakes) as well. An XML schema validation provides some help here, but the editors need more control over the conversion process. To be able to detect all errors, scholars must have the possibility to see the result of the conversion as soon as possible. In the Ficker projects, the application responsible for the conversion is also able to render TEI-XML data as HTML pages.

As one can see, transforming DOCX to TEI using specific designed stylesheets is quite powerful. There are other constraints in DOCX that can be utilized for TEI conversions: font-faces, font-styles, text-colors, the correction-tracking feature, etc. If one would want to utilize these features in DOCX2TEI, developers only need to embed other stylesheets that make use of these features.

Using DOCX, scholars will not have to interact with an XML editor. Since the conversion is mainly determined by the stylesheets, people will not have to worry about choosing the correct TEI annotations, if the stylesheet does not allow different conversion for the same feature.

3 Comparison to *OxGarage* and similar tools

There are also other solutions that are capable of converting DOCX files to TEI. The most renowned tool for this task is surely *OxGarage*,[9] which is maintained by the TEI Consortium itself. *OxGarage* is not only able to convert DOCX to TEI, but reads many different formats and can convert texts, spreadsheets, and even presentations. *OxGarage* also offers a REST-API (TEI Consortium 2020)[10] through which other applications can execute conversions without the user interface of *OxGarage*. This is especially convenient if one wants to use the conversion routines of *OxGarage*[11] programmatically.

Internally, *OxGarage* uses the XSLT stylesheets. It does offer additional functionality of pre- and post-processing files, but essentially we can regard it as an interface for the execution of stylesheets that are provided by the TEI-Consortium. From the perspective of someone who only needs the conversion of DOCX files to TEI, *OxGarage* and DOCX2TEI – and potentially also other tools that convert DOCX files

[9] https://oxgarage2.tei-c.org.

[10] Cf. the README file of the *OxGarage* Github Repository under https://github.com/TEIC/oxgarage.

[11] We acknowledge that *OxGarage* is a much greater conversion tool than simply a DOCX to TEI converter, however, in this article, we will solely concentrate on the DOCX to TEI conversion function of *OxGarage*.

to TEI through XSL transformations – are quite similar. In both cases the content of DOCX files is put into an XML file using TEI vocabulary and the conversion does not aim for a faithful reproduction of a text-page, but for data conversion of text models. However, the main difference lies in the desired workflow and with that the different behavior of the editors using these tools. In our opinion, *OxGarage* is the appropriate tool for scholars who want to get a text with its structure as TEI data quickly. The enrichment of the data (e.g., creating the metadata for the TEI header or the semantic annotations) is done later by using other XML-editors. With *OxGarage*, people can take a "short-cut" in the creation of TEI-data.

DOCX2TEI, on the other hand, aims at the substitution of an XML editor. The conversion creates as clean TEI data as possible without any need of manually editing its results. Scholars and editors use MS *Word* as a tool both for the transcription and the annotation. We can even boldly claim that DOCX2TEI makes MS *Word* a project specific TEI editor. This short comparison shows that *OxGarage* and DOCX2TEI require different efforts from their users. For *OxGarage*, editors need to know the XML rules, XML editors, and the TEI guidelines. For the technicians, on the other hand, the programming efforts are reduced.

We can see that while DOCX2TEI and *OxGarage* are both capable of converting DOCX files to TEI, their user scenarios are quite different. DOCX2TEI only works well if it is customized to meet the needs of a specific project. This means that if another project wants to re-use DOCX2TEI, the chance is very high that the conversion routines must be adapted as goals and focuses of different editions will vary. *OxGarage*, on the other hand, produces uniformed conversion results for a wide range of use cases.[12] If one wants to adapt the *OxGarage* conversion for their own project, the customization must be done *after* the conversion.

What about other tools that are capable of converting DOCX files to TEI data? Since conversion data from DOCX- and ODT-files (i.e. the standard file format for *Open-* and *LibreOffice* text processors) is considered as a standard workflow in academic publishing (McGlone 2014), we can expect that there is a wide range of tools available for this functionality. A *Google*-search reveals that several projects are working on this issue as well.[13] The handling of the conversion may differ (some require users to execute a command in the console, other require a preinstalled software library), but the conversion itself is usually done through XSLT. Thanks to the fact that most of these projects are open-sourced, scholars can adapt them to meet their own need.

---

[12] This is not entirely true. The online version of *OxGarage* has been updated several times and the stylesheets used for data conversion are also improved. However, if we stick to the same version of *OxGarage*, the conversion result will not vary.

[13] For instance *Pandoc* (MacFarlane 2006–2022), or *TEI Publisher* (2020). The *Oxygen XML Editor* can convert DOCX to TEI-XML (SyncRO Soft 2019), too.

Some tools also provide customization features to ease the conversion.[14] Thus, one can use these tools as a quick way to produce preliminary TEI-XML, or one can even use them to make MS *Word* a TEI editor. Furthermore, it should be emphasized here that several digital editing projects[15] have adapted a workflow that takes DOCX files to produce TEI files and consequently MS *Word* can be a viable alternative to XML editors.

4 Insights from working with DOCX2TEI

We have been using DOCX2TEI since 2010, have adapted DOCX2TEI for different projects, and we have learned from our practical experience. In the following section we want to share some insights we gained in the process of developing and improving our workflow. We believe that readers can profit from our report as other project teams using tools similar to DOCX2TEI may have similar challenges we have encountered in the past.

1. The main rule is: editorial guidelines need to be clear and exact! Consequently, the editing experience (whether using an XML editor or other tools like MS *Word*) is more efficient and customizing stylesheets for DOCX2TEI becomes easier too. Just like in printed editions, editorial guidelines will reflect features from the primary source that are to be edited as well as the goal of the edition. If editors have clear guidelines, the editing process itself will become easier, and developers, too, can come up with solutions for transformation and presentation more easily. Of course, every now and then situations come up that are not covered by the editorial guidelines. While a printed book can usually find 'ad hoc solutions', e.g., a deviation from the editorial guidelines, applying these kinds of deviations from the editorial guidelines may also involve the need of changing software code. They are therefore more difficult to deal with when one uses (and is committed to) DOCX2TEI. There might also be situations for which an appropriate solution is not found, in those cases we either forfeit the desired ad hoc solution or we have to manually encode the respective solution.

2. Good and regular communication between scholars and technicians is essential. This is especially obvious when one wants to implement customizations. This

[14] The *Cirilo* client, for example, can use customization to improve the result of its DOCX to TEI conversion, see: http://gams.uni-graz.at/o:gams.doku#custom.

[15] One can find several work reports of digital editing projects that have the DOCX to TEI conversion as subject, e.g. Lehečka (2019), or the following blog post by Dot Porter: http://www.dotporterdigital.org/workflow-ms-word-to-tei/. The software TEI publisher can also ingest DOCX-files and convert them to TEI.

requires scholars stating their intentions clearly (cf. rule 1) to enable the technicians to adapt routines and codes accordingly. This is, however, not the only case where good communication helps to resolve misunderstandings. This can be best achieved simply by talking with each other. Especially at the beginning of a project, meetings of project members should be scheduled more often, e.g. once in two weeks. Later, when people have more routine, these meetings will become less frequent. However, keeping the possibility of individual meetings (e.g. between scholars and technicians) is essential. We have made good experience using a ticket system for documenting problems, concerns, and their solutions.[16] The ticket system (or other online collaborative tools used for documentation purposes) should be openly available for all project members. The methodology proposed by the so-called Agile Software Development (Beck 2001, Dingsoyr et al 2012) can also be helpful, as it emphasizes the importance of embedding users (clients) into the design and development processes.

3. Any problem reported by the users should be taken seriously, but not personally. There can be unusual bug fixes and workarounds. We have experienced that scholars have tried to compensate for programming errors by simply not using specific features. While the programming error was very easy to correct, scholars did not realize that it was something that can be "fixed" (cf. Rule 2). This is truly a remarkable symptom of an endeavor that involves multiple parties. This kind of behavior certainly can be minimized by respecting each member of the editing team and caring about her or his problems.

4. Even when one uses DOCX2TEI, scholars still must acquire some knowledge of TEI. At least they must know the difference between structured metadata and XML data that is semi-structured. They need to understand the difference between data and data representation. We do not encourage using tools like DOCX2TEI without giving scholars a proper introduction to TEI. The reason for this suggestion is that digital editing requires a different, more large-scale understanding of 'edition' than editing a printed volume. In a printed volume, scholars work in terms of printed pages. In a digital edition employing TEI-XML (or any other technical means with which the editors are not familiar), however, scholars must think in terms of data-models that represent the primary source. Furthermore, these models themselves are represented in other data-models to produce HTML views or printed pages. Learning TEI is the best way to grasp these basic concepts.

[16] In our project we use *GitLab* provided by our university for both version control of the source code and for the documentation of the issues within projects.

5. Reusing DOCX2TEI for other projects is doable, even in different settings. In the last six years we have also ported DOCX2TEI to other projects. One of them is the creation of a literature map that is a part of the project *The Tyrol / The South Tyrol – A literary Topography* (FWF P26039, led by Johann Holzner). The aim of the literature map is to create a database that contains references to geographical places in literature. Such a reference could be, e.g., a description of a village in South Tyrol, or a poem written about a mountain peak.

For this project, we have prepared DOCX files so that scholars only have to fill in the data in MS *Word* and upload it to the database. After the conversion (and if the result of the conversion is valid), the TEI data is further processed together with data coming from a separate database on geographical locations, containing data on coordinates coming from *GeoNames*[17] to produce an interactive map that is now accessible in *LiteraturTirol*.[18]

5 Conclusion

This chapter introduced a DOCX to TEI-XML conversion that is successfully used in several projects at the University of Innsbruck. The discussed workflow is particularly well-suited for projects with a large amount of material to be transcribed or material already available as DOCX files. In such cases, a DOCX to TEI-XML workflow can save quite some time, however, the conversion tool plays a central role in the process and the quality of the result depends not only on the edited text in the DOCX file(s), but on the conversion tool and the intermediary stylesheets too. We are quite positive that for certain projects our DOCX2TEI-approach can be more efficient than using other conversion tools, due to its modularity and adaptability to a project's needs. Compared to *OxGarage*, DOCX2TEI shows certain benefits, mainly due to the fact that data enrichment is realized prior to the conversion. However, users must keep in mind that MS *Word* is not a full-blown XML editor, thus in all projects that employ a DOCX to TEI conversion workflow the data modeling is subject to limitations. An edition project that relies solely on DOCX as an editing tool cannot use the full flexibility that TEI provides, and furthermore MS *Word* does not include the options to check wellformedness and validity in accordance with the TEI.

[17] https://www.geonames.org.
[18] https://literaturtirol.at/landkarte.

Bibliography

Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning et al. 2001. "Manifesto for Agile Software Development." https://agilemanifesto.org.

Deutsche Nationalbibliothek, eds. 2019. "Gemeinsame Normdatei (GND)." Last modified June 25, 2019. Accessed September 30, 2019. https://www.dnb.de/DE/Professionell/Standardisierung/GND/gnd_node.html.

Dingsøyr, Torgeir, Sridhar Nerur, VenuGopal Balijepally, and Nils Brede Moe. 2012. "A decade of agile methodologies: Towards explaining agile software development." *The Journal of Systems and Software* 85: 1213–21.

ECMA International, eds. 2016. "Standard ECMA-376. Office Open XML File Formats." Accessed September 30, 2019. http://www.ecma-international.org/publications/standards/Ecma-376.htm.

Hirsch, Waltraud, ed. 2010. "Otto Weininger: Kritische Online-Ausgabe der Werke und Briefe." Charlottesville: Intelex. Accessed September 30, 2019. http://library.nlx.com/xtf/view?docId=weininger_de/weininger_de.00.xml;chunk.id=div.weininger.pmpreface.1;toc.depth=2;toc.id=div.weininger.pmpreface.1;hit.rank=0;brand=default.

Kelly, Peter. 2003. "Microsoft Office System and XML: Bringing XML to the Desktop." Accessed September 30, 2019. https://docs.microsoft.com/en-us/previous-versions/office/developer/office-2003/aa159914(v=office.11)?redirectedfrom=MSDN.

Lehečka, Boris. 2019. "Using Microsoft Word for preparing XML TEI-compliant digital editions." Accessed September 30, 2019. https://zenodo.org/record/3451430#.X7zHUVAxmUk.

MacFarlane, John. 2006–2022. "Pandoc, a Universal Document Converter." Accessed April 1, 2020. https://pandoc.org/index.html.

McGlone, Jonathan, 2014. "Preserving and Publishing Digital Content Using XML Workflows." In *Library Publishing Toolkit*, ed. by Allison P. Brown, IDS Project Press, 97–108.

Porter, Dot. 2018. "Workflow: MS Word to TEI. March 21, 2018." Accessed February 1, 2021. http://www.dotporterdigital.org/workflow-ms-word-to-tei.

SyncRO Soft. 2019. "TEI P5 Document Type (Framework)." Accessed April 1, 2020. https://www.oxygenxml.com/doc/versions/22.0/ug-editor/topics/author-teip5-doc-type.html.

TEI Consortium, eds. 2019. "TEI P5: Guidelines for Electronic Text Encoding and Interchange. Version 3.6.0." Accessed September 30, 2019. http://www.tei-c.org/Guidelines/P5.

TEI Consortium. 2020. "OxGarage." Accessed April 1, 2020. https://oxgarage.tei-c.org. Source code available under https://github.com/TEIC/oxgarage.

TEI Consortium. 2020. "TEI Publisher, the Instant Publishing Toolbox." Accessed April 1, 2020. https://teipublisher.com/exist/apps/tei-publisher/index.html.

W3-Consortium, eds. 2003. "Mathematical Markup Language (MathML) Version 2.0 (Second Edition) W3C Recommendation 21 October 2003." Accessed September 30, 2019. https://www.w3.org/TR/MathML2.

W3-Consortium, eds. 2007. "XSL Transformations (XSLT) Version 2.0. W3C Recommendation 23 January 2007." Accessed September 30, 2019. https://www.w3.org/TR/2007/REC-xslt20-20070123.

W3-Consortium, eds. 2008. "Extensible Markup Language (Xml) 1.0 (Fifth Edition). W3c Recommendation 26 November 2008." Accessed September 30, 2019. https://www.w3.org/TR/2008/REC-xml-20081126.

W3-Consortium, eds. 2009. "Namespaces in XML 1.0 (Third Edition). W3C Recommendation 8 December 2009." Accessed September 30, 2019. https://www.w3.org/TR/REC-xml-names.

W3-Consortium, eds. 2011. "Scalable Vector Graphics (SVG) 1.1 (Second Edition). W3C Recommendation 16 August 2011." Accessed September 30, 2019. https://www.w3.org/TR/SVG11.

Wang, Joseph. 2016. "From DOCX via TEI to Literature Map." Poster. Access on September 30th 2019. http://tei2016.acdh.oeaw.ac.at/sites/default/files/Joseph%20Wang%20From%20DOCX%20via%20TEI%20to%20Literature%20Map.pdf.

Zangerle, Ignaz, Walter Methlagl, Franz Seyr, Anton Unterkircher, and Martin Alber, eds. 1986–1996. *Ludwig Ficker. Briefwechsel*, four volumes. Salzburg: Otto Müller and Innsbruck: Haymon.