Alleviating the Last Mile of Encoding: The *mei-friend* Package for the Atom Text Editor

MEC 2021 BEST PAPER AWARD

Werner Goebl
Department of Music Acoustics –
Wiener Klangstil (IWK), mdw –
University of Music and Performing Arts
Vienna, Austria
goebl@mdw.ac.at

David M. Weigl
Department of Music Acoustics –
Wiener Klangstil (IWK), mdw –
University of Music and Performing Arts
Vienna, Austria
weigl@mdw.ac.at

Abstract

Though MEI is widely used in music informatics and digital musicology research, the relative lack of authoring software and the specialised nature of its community have limited the availability of high-quality MEI encodings. Translating to MEI from other encoding formats, or generating MEI via optical music recognition processes, is thus a typical component of many MEI-project workflows. However, automated translations rarely achieve results of sufficient quality, a problem well-known in the community and documented in the literature. Final correction and validation by hand is therefore a common requirement. In this paper, we present *mei-friend*, an extension to the Atom text editor, which aims to relieve the degree of manual labour required in this process. The tool facilitates most common MEI editing tasks including the insertion and manipulation of MEI elements, makes the encoded score visible and interactively accessible to the user, and provides quality-of-life conveniences including keyboard shortcuts for editing functions as well as intelligent navigation of the MEI hierarchy. We detail the tool's implementation, describe its functionalities, and evaluate its responsiveness during the editing process, even when editing very large MEI files.

Introduction

The manual encoding of non-trivial music scores using MEI currently requires extensive specialist knowledge, both of XML syntax, processes, and workflows more broadly, and of the MEI schema in particular. Even with such expertise in place, the encoding activity is laborious and is usually frequently interrupted to re-render and inspect the notation corresponding to the encoding, typically using the Verovio[1] MEI engraving toolkit.

There are alternatives to manual encoding from scratch, though none afford the same level of control over the quality of the final encoding outcome. Pre-existing encodings in other formats, e.g., MusicXML, or the Humdrum **kern data format, may be identified and translated to MEI, though this is liable to introduce conversion issues; or, optical music recognition (OMR) tools may be used to generate encodings from score images, though these tend to be error-prone and usually do not produce MEI natively; or, scores may be typeset using the graphical user interfaces of commercial notation software, albeit with similar limitations on native MEI-support (though in the case of Sibelius, the Sibmei[2] extension may be used to accomplish this).

In each of these approaches, once the rough process of generating or bootstrapping digital score encodings and converting them to MEI is accomplished, a manual editing phase is required in which the MEI is checked for errors and inconsistencies, polished, and finalised if high-quality encodings are to be achieved.

1 https://www.verovio.org (accessed January 12, 2022).

2 https://github.com/music-encoding/sibmei (accessed January 12, 2022).

Background

Imperfect Nature of Encoding Format Conversion

Challenges of translating between encoding formats are well known in the community and have been the subject of research in recent years:

- [7] surveys the classes of problems arising in conversion between music encodings in **kern, Lilypond, MEI, and MusicXML formats;
- [6] demonstrates how apparently equivalent encodings in different encoding formats can lead to discrepancies in the outcomes of music analyses;
- [8] considers how methodological choices undertaken during the generation and processing of music encodings may impact on conversion success in unintuitive ways.

The problems outlined in these studies can in part be addressed by improving the conversion processes themselves, particularly where whole classes of problems arise from flawed conversion of frequently used encoding constructs. We have worked in this direction by improving Verovio's MusicXML to MEI conversion process and have promoted community involvement by participating in the organisation of a "Developing Verovio" workshop at the 2020 Music Encoding Conference. But even when the conversion process is improved as far as possible, there will be certain aspects benefiting from final manual 'polishing', due to semantic inconsistencies between encoding formats. Conducting such manual MEI finalisation in a text editor is labour intensive. Schema-aware but music-naive XML editors, such as Oxygen, partially alleviate the process by automating validation tasks. Like plain text editors, however, they (somewhat unintuitively) situate all interactions with music notation in the text (XML) domain, placing the cognitive load of navigating and modifying the MEI encodings in a musically-informed way squarely on the user. Specialised MEI editors provide further assistance by incorporating interactive, dynamically-rendered scores as additional means of navigation and interaction and by providing shortcuts to accomplish the most frequent notation-editing operations.

Editing MEI

Verovio's affordances for graphical interaction with encoded music notation [9] have resulted in the creation of several score editors incorporating its JavaScript toolkit to render and manipulate encoded scores.

Neon is a web-based square-notation editor aimed at correcting encodings of neume notation generated via OMR [10]. Alongside support for essential elements of square-notation music, including neumes (*punctums*, *virgas*, *custos*), basic neume groupings (e.g., *pes*, *clivis*, *torculus*), and C and F clefs and custos, there is also built-in support for displaying and editing text (lyrics) and interactive integration of source image facsimiles using the International Image Interoperability Framework (IIIF) [11]. Another editor for MEI encodings of neume notation, *monodi*+ [1], focuses on incorporating editorial variants and capturing hierarchical structures in medieval monophonic music, and it employs a custom rendering engine in place of Verovio. Both editors provide powerful tools for the transcription, validation, and finalisation of MEI encodings, but they are limited to neume notation and consequently do not support the comparatively more complex notation context of CMN.

Verovio-Humdrum-Viewer [12] is an elaborated editor for CMN and mensural notation developed for **kern. It supports a variety of basic editing operations through a graphical user interface, alongside an integrated text editor for more complex changes.[3] It has been used to generate a substantial corpus of encodings, a notable ongoing endeavor being the encoding of the entire Chopin first editions.[4] Beside its native support for **kern, the tool is able to import encodings in MEI and MusicXML and to export encodings to MEI and (subsequently) render them as SVG using Verovio. However, it does not support editing of MEI directly; all editing operations are performed in **kern prior to MEI conversion. This limits the editor's applicability for finalising MEI encodings, as the restriction of the editing functionality to the **kern format means that it cannot be used to address post-MEI-conversion issues or inconsistencies. Further, this division into 'read-write' **kern and 'read-only' MEI introduces addressability issues into the generated MEI encodings. In converting from **kern, the tool mints

3 See documentation at https://doc.verovio.humdrum.org/ (accessed January 12, 2022).

4 https://chopin.nifc.pl/en/chopin/dziedzictwo (accessed January 12, 2022).

new `@xml:id` (local XML identifier) attributes for the created MEI elements according to their corresponding **kern-native line and column numbers. As a consequence, any (**kern) editing operations that affect the number of lines (e.g., adding a new note) will re-assign the identifiers in the corresponding MEI to potentially different elements. While this is not a problem in terms of local addressability within the updated MEI file as the new identifiers retain their internal consistency, it does violate the persistence of identifiers for use in a global context, as required, e.g., in Linked Data applications, where MEI elements may be externally addressed using fragment URIs [16].

Verovio has been integrated into the Atom[5] text editor desktop application using an extension called *mei-tools-atom* [5]. This tool embeds a dynamically updating Verovio score SVG within an editor pane displayed alongside the MEI (XML) text buffer. It offers basic navigation aides, such as clicking a rendered score element to jump to the corresponding line of MEI or selecting a line of MEI to highlight the corresponding rendered score element on the current page. However, its functionality is otherwise limited, the integrated version of Verovio is outdated, the code has not been updated for two years, and the GitHub code repository of the project has recently been archived (set to read-only) by its owner.

Finally, a prototype editor has been built into the Verovio toolkit. This editor has the advantage of being a native part of Verovio, requiring no other dependencies, and supporting the development of Web applications that interact directly with MEI encodings through the Verovio toolkit[6] via `vrvTk.edit({editCommand})` function calls. However, only a small set of basic edit commands (such as inserting and deleting notes, and modifying or setting attributes of existing elements) is currently supported, with many functionalities important in the MEI finalisation process yet to be implemented.

The *mei-friend* Package for the Atom Text Editor

Use Case: Encoding Beethoven's Piano Works

TROMPA (Towards Richer Online Music Public-domain Archives) is a project building on the semantic affordances of MEI encodings to address the music information needs of wider audiences, with distinct project strands focusing on music performers and enthusiasts, as well as on scholars [15]. The facilitation of the MEI encoding process has been one focus of research in this project, as the relatively small number of available, publicly licensed encodings (relative to TROMPA's target repertoire of European classical music) necessarily limits the applicability of the developed user-facing prototypes.

One avenue of research has been into the crowd-generation of MEI by parcelling up the process of validating and fixing OMR-generated encodings into smaller (e.g., measure-level) units and farming these out to larger communities. This approach is relevant where a larger group of music experts share a time-sensitive interest in the generation of a new encoding and thus was developed with a focus on orchestra use cases [3]. TROMPA's development of this approach is partly informed by the OpenScore Initiative [2] which has applied comparable crowd-based processes in the generation of MSCX (MuseScore) format score encodings.

However, such crowdsourced encodings still benefit from a finalisation step, and other use-case contexts do not necessarily include ready access to a motivated crowd and so stand to benefit all-the-more from a facilitation of the encoding process by individuals using an MEI editor.

To illustrate our requirements, we briefly describe the generation of solo-piano encodings for TROMPA's instrumental players use-case [13, 14]. The recent 250th birthday of Ludwig van Beethoven and the available encodings of his 32 sonatas for piano in the **kern format[7] initially motivated the attempt to encode the remaining piano solo works by Beethoven into MEI format. These pieces comprise several variation cycles (among them the *Eroica Variations,* Op. 35, the *Diabelli Variations,* Op. 120, or the C-minor Variations, WoO 80), Rondos (Op. 51), Bagatelles (Opp. 33, 119, 126), the Polonaise (Op. 89), Sonatinas (WoO Anh. 5), Phantasie (Op. 77), and pieces, such as Rondo a Capriccio, Op. 129, and Clavierstück "Für Elise", WoO 59. As the target edition to encode, we chose Breitkopf and Härtel's first complete edition from 1862–1868, fully available as scanned,

5 https://github.com/atom/atom (accessed January 12, 2022).

6 Demonstration available at https://editor.verovio.org (accessed January 12, 2022).

7 https://github.com/craigsapp/beethoven-piano-sonatas (accessed January 12, 2022).

publicly licensed images at IMSLP.[8] The aim of the encoding was to follow the target edition as closely as possible using the current version 4.0.1 of MEI as engraved by Verovio (currently version 3.3). The corpus encoded now comprises 18 works by Beethoven spanning over 220 pages, all published under open license on GitHub.[9]

In encoding this corpus, we began by scanning a modern commercial edition of each score using a Konica Minolta C308 printer at 300dpi grayscale at A3 landscape with the saturation set 2 steps darker; this provided considerably better OMR results than when we started with the publicly licensed PDFs of the target edition directly. The derived PDFs were rotated, and black margins cropped, using Adobe Acrobat Pro (Version 9). The proprietary Neuraton PhotoScore application (Version 2018 8.8.7) was used to import the rotated and cropped PDFs at highest resolution and to perform OMR. Evident OMR errors, such as incorrect key signatures, meter indications, triplets, or notes, were fixed in-application, and the outcomes were exported to uncompressed MusicXML format. The resulting files were imported into MuseScore (Version 3.1.22425) to clean up further OMR errors and to adjust the encoding to the target edition.

Subsequently, the encodings were re-exported as MusicXML and converted to MEI using Verovio. During this step, we developed fixes for several classes of conversion error, including: support for ending elements, improvements in handling slurs, ties, hairpins, clef changes, cross-staff notes, pedals, arpeggios, turns, and backup elements. These systematic errors were addressed at source through contributions to Verovio's open-source C++ codebase. The MEI encodings were then processed algorithmically to remove duplicated `@accid` and `@accid.ges` attributes, insert missing `@xml:id`s, and to renumber the measures, taking into account measures that do not conform to the meter signature (`@metcon="false"`) and measures inside multiple endings.

Having arrived at this stage, the MEI encodings were largely complete, correct, and faithful to the target edition; but further laborious steps were required to undertake final editing of details, such as inserting slurs and/or precisely placing their beginnings and endings, and adjusting their direction of curvature, inserting or fixing dynamics markings, directives, ornaments, pedal markings, etc. To illustrate: The insertion of a slur requires the `@xml:id` attributes of the target notes at two distinct positions in an encoding to be identified and a line of code to be added to the end of the measure containing the first note or chord. Manually, this operation is quite time consuming and repetitive, as slurs are often misplaced or completely missed during the OMR workflow described above.

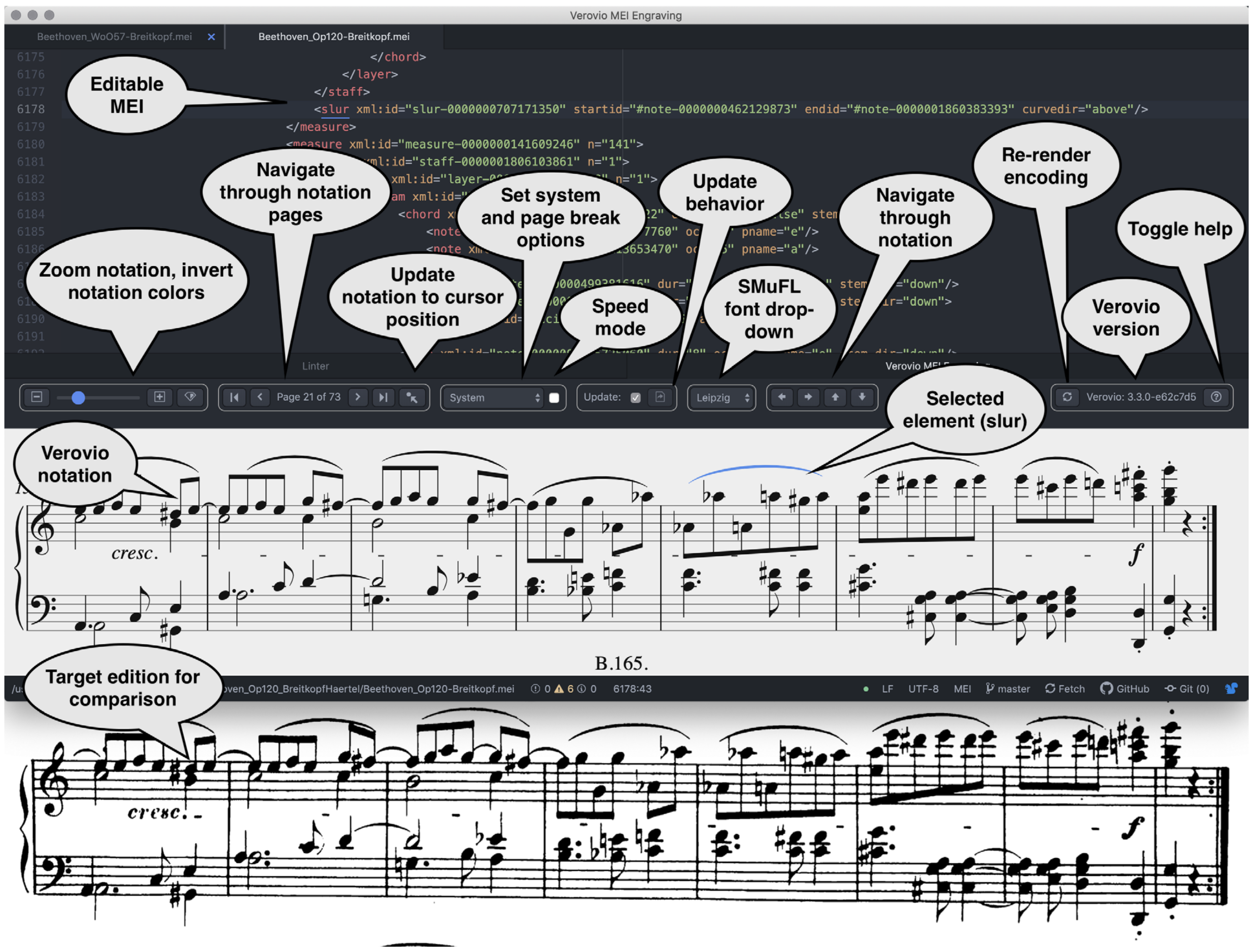
Implementation of *mei-friend*

Specialised MEI editing software was required to ease this finalisation process, but none of the currently available editors (see *Editing MEI*) fulfilled the requirements of our use-case for CMN notation. We gratefully made use of the existing open-source (MIT-licensed) code for *mei-tools-atom*, forking it with a new name (to reflect the new project direction) and housing it in a new GitHub repository [4] under the same license.

mei-friend is written in JavaScript using the Atom package framework and automatically loads the latest release version of Verovio as a dependency using the Node package manager (npm). Through the provided `text-Editor` object, *mei-friend* interacts with Verovio's JavaScript toolkit to ingest the text buffer (MEI) content and to render the corresponding SVG, displaying it in a repositionable panel within the Atom application window, alongside the panel displaying the text buffer. User interactions with both panels are supported, allowing, e.g., for the selection of a line of text to highlight the corresponding rendered score element(s), and vice versa. Atom also provides tight git integration, supporting easy and fast interaction with MEI encodings hosted on GitHub.

8 https://imslp.org/wiki/Ludwig_van_Beethovens_Werke_(Beethoven,_Ludwig_van) (accessed January 12, 2022).

9 https://github.com/trompamusic-encodings/ (accessed January 12, 2022).

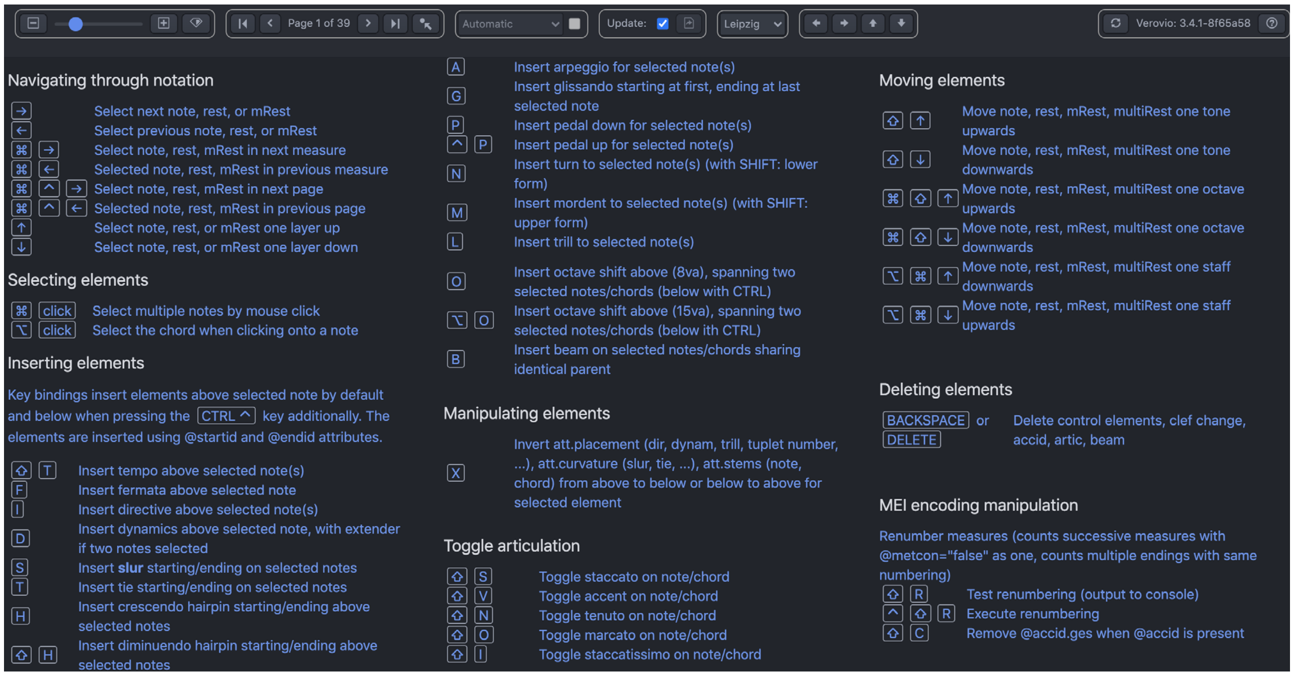

Verovio MEI Engraving
Beethoven_WoO57-Breitkopf.mei
Beethoven_Op120-Breitkopf.mei
Editable MEI
Navigate through notation pages
Set system and page break options
Update behavior
Navigate through notation
Re-render encoding
Toggle help
Zoom notation, invert notation colors
Update notation to cursor position
Speed mode
SMuFL font drop-down
Verovio version
Linter
Page 21 of 73
System
Update:
Leipzig
Verovio: 3.3.0-e62c7d5
Selected element (slur)
Verovio notation
cresc.
B.165.
Target edition for comparison
ven_Op120_BreitkopfHaertel/Beethoven_Op120-Breitkopf.mei
6178:43
LF UTF-8 MEI master Fetch GitHub Git (0)
cresc.

```
                          </chord>
                       </layer>
                    </staff>
                    <slur xml:id="slur-0000000707171350" startid="#note-0000000462129873" endid="#note-0000001860383393" curvedir="above"/>
                 </measure>
                 <measure xml:id="measure-0000000141609246" n="141">
```

Figure 1: Annotated screenshot of the *mei-friend* package for the Atom text editor, showing an excerpt of the encoding of Beethoven's *Diabelli Variations* (top), its automatic rendering with Verovio (middle) and the same system in the target edition by Breitkopf & Härtel (bottom, another window outside the Atom editor).

Figure 1 illustrates the actions for display, navigation, and editing available to the user through the main menu bar (from left to right):

- viewing controls to scale the score in the notation panel and to invert the notation colours (black on white, or white on black);
- controlling page turning (jump to first, previous, next and last page) and flipping the score page to correspond to the current cursor position in the text buffer;
- setting Verovio's `--breaks` option, and toggling between normal and speed mode;
- setting the updating (score re-rendering) behaviour: *automatic*, after each edit to the text buffer; or, *manually*, upon clicking the 'update' button;
- selecting SMuFL font for notation;
- navigating through the notation by modifying the currently-selected note or rest, via button clicks or key-bindings (note-wise back-/forwards, measure-wise back-/forwards, page-wise back-/forwards, layer-/staff-wise up/down);
- re-processing (loading and serialising) the MEI XML text-buffer contents through Verovio to standardise the order of encoded elements and insert missing `@xml:ids` (or remove unnecessary `@xml:ids`);
- displaying Verovio's current version, and the key bindings help panel (see Figure 2).



Page 1 of 39 · Automatic · Update: ✓ · Leipzig · Verovio: 3.4.1-8f65a58

Navigating through notation

→	Select next note, rest, or mRest
←	Select previous note, rest, or mRest
⌘ →	Select note, rest, mRest in next measure
⌘ ←	Selected note, rest, mRest in previous measure
⌘ ^ →	Select note, rest, mRest in next page
⌘ ^ ←	Selected note, rest, mRest in previous page
↑	Select note, rest, or mRest one layer up
↓	Select note, rest, or mRest one layer down

Selecting elements

⌘ click	Select multiple notes by mouse click
⌥ click	Select the chord when clicking onto a note

Inserting elements

Key bindings insert elements above selected note by default and below when pressing the CTRL ^ key additionally. The elements are inserted using @startid and @endid attributes.

⇧ T	Insert tempo above selected note(s)
F	Insert fermata above selected note
I	Insert directive above selected note(s)
D	Insert dynamics above selected note, with extender if two notes selected
S	Insert **slur** starting/ending on selected notes
T	Insert tie starting/ending on selected notes
H	Insert crescendo hairpin starting/ending above selected notes
⇧ H	Insert diminuendo hairpin starting/ending above selected notes
A	Insert arpeggio for selected note(s)
G	Insert glissando starting at first, ending at last selected note
P	Insert pedal down for selected note(s)
^ P	Insert pedal up for selected note(s)
N	Insert turn to selected note(s) (with SHIFT: lower form)
M	Insert mordent to selected note(s) (with SHIFT: upper form)
L	Insert trill to selected note(s)
O	Insert octave shift above (8va), spanning two selected notes/chords (below with CTRL)
⌥ O	Insert octave shift above (15va), spanning two selected notes/chords (below ith CTRL)
B	Insert beam on selected notes/chords sharing identical parent

Manipulating elements

X	Invert att.placement (dir, dynam, trill, tuplet number, ...), att.curvature (slur, tie, ...), att.stems (note, chord) from above to below or below to above for selected element

Toggle articulation

⇧ S	Toggle staccato on note/chord
⇧ V	Toggle accent on note/chord
⇧ N	Toggle tenuto on note/chord
⇧ O	Toggle marcato on note/chord
⇧ I	Toggle staccatissimo on note/chord

Moving elements

⇧ ↑	Move note, rest, mRest, multiRest one tone upwards
⇧ ↓	Move note, rest, mRest, multiRest one tone downwards
⌘ ⇧ ↑	Move note, rest, mRest, multiRest one octave upwards
⌘ ⇧ ↓	Move note, rest, mRest, multiRest one octave downwards
⌥ ⌘ ↑	Move note, rest, mRest, multiRest one staff downwards
⌥ ⌘ ↓	Move note, rest, mRest, multiRest one staff upwards

Deleting elements

BACKSPACE or DELETE	Delete control elements, clef change, accid, artic, beam

MEI encoding manipulation

Renumber measures (counts successive measures with @metcon="false" as one, counts multiple endings with same numbering)

⇧ R	Test renumbering (output to console)
^ ⇧ R	Execute renumbering
⇧ C	Remove @accid.ges when @accid is present

Figure 2: The *mei-friend* help panel lists the implemented key bindings for navigating through the notation, selecting elements, and performing several basic insert operations (control elements, such as slurs, hairpins and ornaments) and manipulating them (inverting the placement, curve direction, or stem direction of multiple selected elements). Bindings illustrated for Mac OSX; adjusted to native OS-conventions on other platforms.

The tool's editing capabilities currently comprise inserting control elements (slur, tie, hairpin, dynamics, directive, tempo indication, fermata, trill, mordent, turn, arpeggio, glissando, pedal, octave) or layer elements (beam) on selected elements and manipulating some of their properties (curve direction, placement, and stem direction). This is achieved using specialised keyboard shortcuts as listed in the help panel (Figure 2). The underlying logic is to select one or two notes and to use a keyboard shortcut to insert new control elements into the MEI code. For example, pressing the `S` key with two selected notes inserts a slur at the end of the measure containing the first selected note, the slur spanning from the first to the second selected note. Subsequent key-presses of X toggle the placement (`@curvedir`) of the slur from automatic (attribute not set) to "above" to "below".

Combining the keyboard shortcut with the `CTRL` key inserts the control element with the `@curvedir` attribute pre-set to "below". There are specialised key-bindings for other elements: e.g., crescendo versus diminuendo hairpins, or starting versus ending pedal markings. Some insertion operations are more complex: for example, when inserting an octave shift element spanning two selected notes in the same staff, the `@oct.ges` attributes of all notes falling between the two selected will be modified accordingly to ensure valid MEI (and will be reset when a selected octave element is deleted). The complete list of key bindings is given in Figure 2 showing the help panel of *mei-friend*.

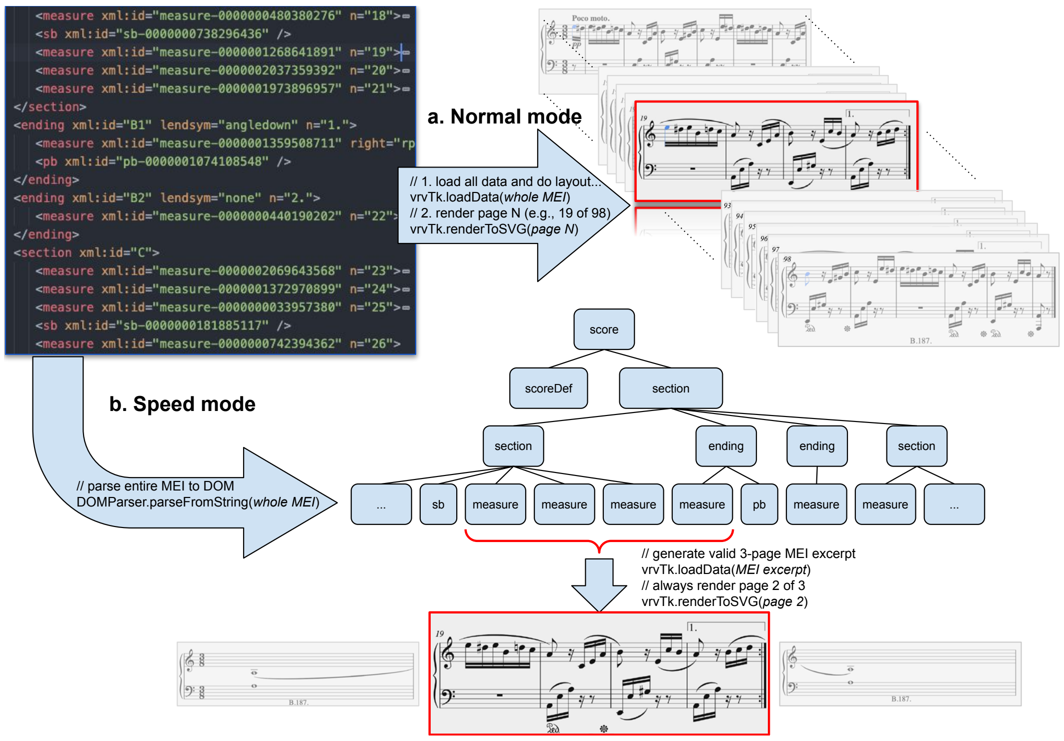
Further functionalities include:

- toggling of articulation elements (staccato, marcato, ...) on selected notes, chords, or higher-level units containing those, such as beams. Following the logic mentioned above, the placement of articulations can be inverted by pressing the `X` key;
- deleting inserted elements with the `DELETE` or `BACKSPACE` key. Currently supported are all control elements, beam, clef change, accidentials, and articulation, moving elements up and down in pitch changing the `@pname`/`@oct` for notes or the `@ploc`/`@oloc` attributes for rests (including `mRest` and `multiRest`), and moving elements up and down staffwise to realise cross-staff notation of piano scores;
- support for specific encoding manipulations, such as re-assigning measure numbers, which are liable to be misattributed during OMR and encoding conversion processes where incomplete measures may be missed, or removing doubled `@accid.ges` attributes when an `@accid` attribute is present.

Beside supporting control element insertion, *mei-friend* improves upon *mei-tools-atom* by supporting: better dependencies management (now loading the latest release version of Verovio via the Node package manager, and showing the Verovio version number in the user interface to help diagnose conversion issues), user interface improvements (introduction of tooltips, a help panel and documentation, keyboard shortcuts, grouping of icons, and navigation inside the notation panel through buttons or key-bindings), and facilitated interaction with Verovio (setting breaks options; selecting SMuFL font; re-processing the MEI XML through Verovio to provide consistent formatting, fix ordering, and generate any missing `@xml:ids`).

Performance Considerations

Large MEI encodings (of about 25,000 lines and more) cause performance issues with Verovio, affecting both *mei-tools-atom* and *mei-friend* (in its default "normal mode"). This is because the entire MEI encoding must be parsed and laid out by Verovio on import before the rendered SVG for a given requested page can be returned for display in the user interface (see Figure 3a). As the current implementations of both Atom extensions are single threaded, the corresponding time-intensive loading and re-rendering operations make the user interface increasingly unresponsive with larger files. When editing the MEI directly in the text buffer, automatic updating of the rendered notation involves repeating this intensive process after each edit. To circumvent this issue, automatic updates may be disabled in *mei-friend*'s interface, instead allowing the user to request re-rendering of the notation manually on button click.



```
    <measure xml:id="measure-0000000480380276" n="18">
    <sb xml:id="sb-0000000738296436" />
    <measure xml:id="measure-0000001268641891" n="19">
    <measure xml:id="measure-0000002037359392" n="20">
    <measure xml:id="measure-0000001973896957" n="21">
</section>
<ending xml:id="B1" lendsym="angledown" n="1.">
    <measure xml:id="measure-0000001359508711" right="rp
    <pb xml:id="pb-0000001074108548" />
</ending>
<ending xml:id="B2" lendsym="none" n="2.">
    <measure xml:id="measure-0000000440190202" n="22">
</ending>
<section xml:id="C">
    <measure xml:id="measure-0000002069643568" n="23">
    <measure xml:id="measure-0000001372970899" n="24">
    <measure xml:id="measure-0000000033957380" n="25">
    <sb xml:id="sb-0000000181885117" />
    <measure xml:id="measure-0000000742394362" n="26">
```

```
// 1. load all data and do layout...
vrvTk.loadData(whole MEI)
// 2. render page N (e.g., 19 of 98)
vrvTk.renderToSVG(page N)
```

```
// parse entire MEI to DOM
DOMParser.parseFromString(whole MEI)
```

```
// generate valid 3-page MEI excerpt
vrvTk.loadData(MEI excerpt)
// always render page 2 of 3
vrvTk.renderToSVG(page 2)
```

Figure 3: Processing flow chart from encoding to notation engraving. a) In "normal mode", the entire content of the encoding is passed to the Verovio toolkit. After the notation layout for the entire encoding is processed, a selected page is rendered as SVG. b) In "speed mode", the MEI encoding is loaded into the Document Object Model (DOM). To render the requested page, its corresponding MEI elements are extracted from the DOM and used to generate an artificial MEI encoding with dummy pages placed around the requested page that incorporate single-note 'anchors' for spanning elements. Verovio is then used to render the requested (middle) page.

To further ease the loading and reprocessing requirements posed by large files, we have implemented a "speed mode" (see Figure 3b), which currently operates only with encodings containing break elements (page and system breaks) and requires a correspondingly set `--breaks` option in the interface. Instead of requiring the entire MEI encoding to be processed, speed mode only transfers a single-page excerpt encompassing the most recently selected element in the text buffer to Verovio. Two additional 'dummy' MEI pages are generated to surround the MEI excerpt to be displayed. The preceding page contains a score definition that is updated to the latest clef, meter, and key signature change preceding the MEI excerpt. Both dummy pages each contain a measure with staff and layer elements, and a single note from and to which time spanning elements (such as slurs and hairpins) are connected, so that the middle page (corresponding to the excerpt) is able to show these elements correctly. These excerpting and inserting operations are performed on an XML DOM (Document Object Model) representation of the MEI encoding, allowing them to be processed very efficiently. In this mode, only a constantly small portion of the MEI encoding (the excerpt) is transferred to Verovio, limiting its processing load and thus keeping interactions swift and smooth.

We evaluated the performance using Atom's internal Chromium profiler by measuring the difference in re-processing time (i.e., user interface unresponsiveness) to quantify the increase in performance of our "speed mode" implementation, using an MEI encoding of Beethoven's WoO 57 (Andante favori, 11.8k lines of MEI) and two artificial encodings (50k and 100k lines of MEI),[10] which we generated by adding the `mdiv` elements of several pieces together. Table 1 displays the median amount of time (in ms) taken to i. open the file, ii. flip to the next page, iii. flip to the last page, and iv. insert a slur, in normal and speed mode, calculated across three repetitions on a 2017 iMac with a 4.2 GHz Intel Core i7 running OS X 10.14.6, Atom 1.54.0, and *mei-friend* 0.3.3. Results demonstrate a significantly improved performance and user experience in speed mode working with large MEI encodings when performing initial loading of the file and re-loading after edits. Page flipping was slightly slower, performance deteriorating with later page numbers, as more DOM processing is required to track score definitions.

Known issues and limitations of speed mode remaining to be addressed include potential inconsistencies with normal mode when rendering automatic curvature directions for slurs spanning across the excerpted measure and not rendering time spanning elements using time stamps rather than start-/endids. Speed mode can be enabled and disabled using a checkbox.

	WoO 57 (11.8k lines)	**Beethoven (50k lines)**	**Beethoven (100k lines)**
Normal mode	666 / 37 / 40 / 625 ms	2290 / 30 / 35 / 2301 ms	6760 / 31 / 36 / 6870 ms
Speed mode	143 / 60 / 93 / 71 ms	204 / 38 / 142 / 45 ms	322 / 42 / 239 / 48 ms

Table 1: Evaluation of *mei-friend* performance in normal mode and speed mode for i. opening file / ii. flipping to the next page / iii. flipping to the last page, and / iv. inserting a slur in different MEI-encoding-size contexts.

Planned Functionalities and Future Development

Several improvements, partly requested by users through GitHub issues, have been implemented since the initial submission of this article. Further potential developments include support for speed mode in all break options, and porting this package to the Visual Studio Code editor,[11] which tends to perform better than Atom and includes a similar JavaScript extension mechanism. Ultimately, we envision an integration of the functionalities presented here with browser-based crowdsourcing mechanisms along the lines of those developed within the TROMPA project.

Conclusion

Although MEI is increasingly adopted for use in music edition and scholarship, the lack of native support by commercial notation software has limited the extent of the available MEI-encoded repertoire. While the *mei-friend* package we have presented here is unlikely to suddenly open up MEI encoding to the masses (it still requires extensive technical expertise from its users, both in XML more broadly and in the MEI schema specifically), it does significantly alleviate the burdens of the final stages of MEI encoding. As most typical processes used for generating MEI encodings employ stages in their workflows liable to introduce errors or deviations that require manual intervention, we hope that *mei-friend* will contribute toward making validation and fixing of the corresponding process outcomes less painful and more enjoyable.

10 Available from https://github.com/trompamusic/mei-friend/tree/master/eval (accessed January 12, 2022).

11 https://github.com/Microsoft/vscode (accessed January 12, 2022).

We must acknowledge limitations in the work presented here. The project context that gave rise to *mei-friend* has been characterised by a predominant focus on European classical piano music, particularly Beethoven's works (among others). It is likely that functionalities may have been identified as useful in other encoding contexts, but are currently missing because they simply didn't arise in our use. We hope to collectively work with the MEI community on the continued development of *mei-friend* to further increase its usefulness in the future.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme "Study European heritage, memory, identity, integration and cultural interaction and translation, including its representations in cultural and scientific collections, archives and museums, to better inform and understand the present by richer interpretations of the past" (H2020-EU.3.6.3.1.) under grant agreement No. 770376. We thank Sienna M. Wood and the nCoda team for contributing the *mei-tools-atom* package, and Laurent Pugin for his valued contributions and advice in the development of *mei-friend*'s integration of Verovio.

Works Cited

[1] Eipert, Tim, Felix Herrmann, Christoph Wick, Frank Puppe, and Andreas Haug. "Editor Support for Digital Editions of Medieval Monophonic Music" in *Proceedings of the 2nd International Workshop on Reading Music Systems* (WoRMS 2019), 4–7, https://sites.google.com/view/worms2019/proceedings.

[2] Gotham, Mark, Peter Jonas, Bruno Bower, William Bosworth, Daniel Rootham, and Leigh VanHandel. "Scores of Scores: An OpenScore Project to Encode and Share Sheet Music" in *Proceedings of the 5th International Conference on Digital Libraries for Musicology* (DLfM 2018), 87–95, https://doi.org/10.1145/3273024.3273026.

[3] Linssen, David, Cynthia C. S. Liem, Ioannis Petros Samiotis, and Gonneke de Jong. "TROMPA Deliverable 6.4.2: Working Prototype for Orchestras". Technical report, 2021, https://trompamusic.eu/deliverables/TR-D6.4-Working_Prototype_for_Orchestras_v2.pdf.

[4] *mei-friend* repository, https://atom.io/packages/mei-friend (accessed January 12, 2022).

[5] *mei-tools-atom* repository, https://github.com/nCoda/mei-tools-atom/ (accessed January 12, 2022).

[6] Nápoles López, Néstor, Gabriel Vigliensoni, and Ichiro Fujinaga. "Encoding Matters" in *Proceedings of the 5th International Conference on Digital Libraries for Musicology* (DLfM 2018), 69–73, https://doi.org/10.1145/3273024.3273027.

[7] Nápoles López, Néstor, Gabriel Vigliensoni, and Ichiro Fujinaga. "The Effects of Translation between Symbolic Music Formats: A Case Study with Humdrum, Lilypond, MEI, and MusicXML" presented at the *Music Encoding Conference* (MEC 2019), University of Vienna, Austria, May 29–June 1, 2019, https://music-encoding.org/conference/abstracts/abstracts_mec2019/The%20effects%20of%20translation%20between%20the%20Humdrum%20%20Lilypond%20%20MEI%20%20and%20MusicXML.pdf.

[8] Parada-Cabaleiro, Emilia, and Álvaro Torrente. "Preventing Conversion Failure across Encoding Formats: A Transcription Protocol and Representation Scheme Considerations" in *Music Encoding Conference Proceedings* (MEC 2020), 105–107, https://doi.org/10.17613/etwb-m434.

[9] Pugin, Laurent. "Interaction Perspectives for Music Notation Applications" in *Proceedings of the 1st International Workshop on Semantic Applications for Audio and Music* (SAAM 2018), 54–58, https://doi.org/10.1145/3243907.3243911.

[10] Regimbal, Juliette, Zoé McLennan, Gabriel Vigliensoni, Andrew Tran, and Ichiro Fujinaga. "Neon2: A Verovio-based Square Notation Editor" presented at the *Music Encoding Conference* (MEC 2019), University of Vienna, Austria, May 29–June 1, 2019, https://music-encoding.org/conference/abstracts/abstracts_mec2019/Neon2.pdf.

[11] Regimbal, Juliette, Gabriel Vigliensoni, Caitlin Hutnyk, and Ichiro Fujinaga. "IIIF-based Lyric and Neume Editor for Square-Notation Manuscripts" in *Music Encoding Conference Proceedings* (MEC 2020), 15–18, https://doi.org/10.17613/d41w-n008.

[12] Sapp, Craig. "Verovio Humdrum Viewer" presented at the *Music Encoding Conference* (MEC 2017), Tours, France, May 16–19, 2017. Slides available from http://bit.ly/mec2017-vhv. Tool available from http://verovio.humdrum.org/.

[13] Weigl, David M., and Werner Goebl. "Rehearsal Encodings with a Social Life" in *Music Encoding Conference Proceedings* (MEC 2020), 51–53, https://doi.org/10.17613/5ae5-8387.

[14] Weigl, David M., and Werner Goebl. "TROMPA Deliverable 6.5-2: Working Prototype for Instrumental Players." Technical report, 2021, https://trompamusic.eu/deliverables/TR-D6.5-Working_Prototype_for_Instrument_Players_v2.pdf. Demonstration available at https://clara.trompamusic.eu.

[15] Weigl, David M., Werner Goebl, Tim Crawford, Aggelos Gkiokas, Nicolas F. Gutierrez, Alastair Porter, Patricia Santos, Casper Karreman, Ingmar Vroomen, Cynthia C. S. Liem, Álvaro Sarasúa, and Marcel van Tilburg. "Interweaving and Enriching Digital Music Collections for Scholarship, Performance, and Enjoyment" in *Proceedings of the 6th International Conference on Digital Libraries for Musicology* (DLfM 2019), 84–88, https://doi.org/10.1145/3358664.3358666.

[16] Weigl, David M., and Kevin R. Page. "A Framework for Distributed Semantic Annotation of Musical Score: 'Take it to the bridge!'" in *Proceedings of the 18th International Society for Music Information Retrieval Conference* (ISMIR 2017), 221–228, https://archives.ismir.net/ismir2017/paper/000190.pdf.