Meico: A Converter Framework for Bridging the Gap between Digital Music Editions and its Applications

Axel Berndt
Center of Music and Film Informatics
University of Music Detmold
Detmold, Germany
berndt@hfm-detmold.de

Simon Waloschek
Center of Music and Film Informatics
University of Music Detmold
Detmold, Germany
waloschek@hfm-detmold.de

Aristotelis Hadjakos
Center of Music and Film Informatics
University of Music Detmold
Detmold, Germany
hadjakos@hfm-detmold.de

ABSTRACT

MEI, the established representation format for digital music editions, barely finds consideration in other music-related communities. Reasons are the format's complexity and ambiguity that make processing expensive and laborious. On the other hand, digital music editions are an invaluable source of symbolic music data and further accompanying information far beyond the typical metadata found in other formats. With meico, we provide a novel tool that makes access, processing and use of MEI encoded music more convenient and appealing for other application scenarios. Meico is a converter framework that translates MEI data to a series of formats relevant to many other applications. With ScoreTube we demonstrate this in an audio-to-score alignment scenario.

CCS CONCEPTS

• **Applied computing** → **Sound and music computing**; **Format and notation**; • **Information systems** → *Information retrieval*;

KEYWORDS

Music Encoding Initiative, Digital Music Editions, Format Converter

ACM Reference Format:
Axel Berndt, Simon Waloschek, and Aristotelis Hadjakos. 2018. Meico: A Converter Framework for Bridging the Gap between Digital Music Editions and its Applications. In *Audio Mostly 2018: Sound in Immersion and Emotion (AM'18), September 12–14, 2018, Wrexham, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3243274.3243282

1 DIGITAL MUSIC EDITIONS, MEI AND MIR

Historical-critical music editions typically present one "edited, definitive text" and a critical report that discusses variations between different manuscript or printed sources used by the editor and the editorial decisions he or she made. *Digital* music editions go a step further. They can encode all versions and variants in the edited text, include digitized autographs, manuscripts or prints and create concordances, i.e. cross-references between similar musical sections in different sources. The critical report is normally transformed to a set of annotations that are linked directly to the items to be discussed. Thereby, the process of finding a definitive text becomes transparent. The reader (e.g. musician) can compile a personal definitive text (so-called reading)—even a musical text that differs substantially from the editor's preference—, render and print the corresponding sheet music from this. Digital music editions may even become a format or mode of music research beyond its traditional focus on music notation as it allows more versatile analytic access to its data and various forms of transformation. The project Beethovens Werkstatt, for instance, makes an in-depth investigation of the composition practice of Ludwig van Beethoven by editing selected works in different stages of development [5].

The established representation format for digital music editions is MEI, an XML standard defined by the Music Encoding Initiative [8]. The purpose of MEI is the encoding of musical documents. It covers the logical, gestural, visual, and analytical domain:

> "[...] the *logical* domain includes the musical content or structure including pitches, time values, articulations, dynamics, and all other elements—defined as the symbols that communicate the composer's intentions. The *gestural* domain relates to a performed interpretation of the logical domain (i.e., it encodes information that may be added by a performer [...]). The *visual* domain describes the contributions of an editor, engraver, or typesetter, and encodes information about the physical appearance of the score, such as symbol locations, page layout, or font. Finally, the *analytical* domain covers commentary and analysis of the music document in any of the three previous domains." [13]

The attempt to meet the requirements of all four domains makes MEI a very extensive and complex format so that even the most recent MEI schema definition (v3.0.0) does not cover it entirely. In addition, digital music editions do not have to address all four domains. Many music editions focus on the visual domain. In this case, the MEI encoding describes sheet music documents but not necessarily provides a complete or unambiguous encoding of the logical domain. Examples can be found in the official MEI Sample Encodings. The fact that one and the same information can be encoded in different fashions further increases the format's complexity. Information about pitch, duration, key, and transposition are not always located directly at the note or rest element but can be scattered over the XML tree. Logical and gestural information may complement each other or compete. Resolving ambiguity and complex or unorthodox situations, such as overfull and "underfull" measures, tuplets within tuplets, nested repetitions, da capi, and even more complex sequencing instructions, makes MEI processing

AM'18, September 12–14, 2018, Wrexham, United Kingdom

ACM ISBN 978-1-4503-6609-0/18/09…$15.00
https://doi.org/10.1145/3243274.3243282

an intricate task and probably the biggest obstacle for potential applications to support the format. In this, we see a major reason for its rare support outside the digital music edition community.

However, there are striking reasons to take this hurdle, esp. for the Music Information Retrieval (MIR) community. Digital music editions offer an invaluable combination of symbolic music data and additional information far beyond the typical metadata found in other formats. All this is usually based on musicological research and features an accordingly high scientific quality. Digital music editions motivate new interesting research questions highly relevant to MIR and the demand to gain deeper insight into subjects such as composition styles, performance practices, historical change processes in music tradition, and how all these reflect in the musical works edited. In this, MIR can make valuable contributions to musicology, for instance by providing tools to work on large corpora of MEI encoded music. Further application scenarios include digital music stand technology [7], music notation and music production [17, 19].

Bridging the gap between digital music editions and its applications requires tools that allow easy access and processing of MEI data in the corresponding context. Therefore, we introduce the MEI converter framework *meico* (section 2) that will, in most applications, mark the starting point for further processing. A typical example for this is ScoreTube (section 3) that demonstrates the whole audio-to-score alignment pipeline incl. score rendering and its alignment with audio recordings. It may serve other projects as blueprint for how to include digital music editions in their process. Related work of the respective subjects will be discussed in the corresponding sections.

2 MEICO: MEI CONVERTER

Processing of MEI data typically involves a relatively extensive preprocessing step. Relevant information are extracted and restructured, ambiguity must be resolved, the timing of all musical events is computed and so forth. Some exemplary challenges of this preprocessing are described above. From a developer's point of view this is a profound impediment to accessibility and processability of MEI documents compared to other music formats. From this situation arises the demand for appropriate converters that transform MEI data to formats that are better suited for specific applications. This is the motivation behind the development of *meico*.

Meico is not the first attempt to an MEI converter. In 2007, Billam published his Perl script *mei2midi* [4] to convert MEI to MIDI. He states that "it should be considered an early release, but is already out of date" [4], as MEI underwent some fundamental revisions and sacrificed backward compatibility at this time. A more recent attempt to making MEI data more accessible to the MIR community is the MEI module for music21 [1] which started about the same time as meico's development.

A project that is currently very active is Pugin's Verovio[1] [15], a music typesetting application that reads MEI data, renders SVG scores, features basic MIDI export and good scripting possibilities. Verovio can also read MusicXML input and generate MEI from it. Likewise, Hankinson & Weber's Sibelius to MEI plugin [10]—an MEI exporter for Sibelius—can be utilized to convert any of the supported input formats of the Sibelius music notation software to MEI format.

The Music Encoding Initiative provides a set of XSLT style sheets for MEI conversion [9]. Supported formats are MusicXML, MARC 21, MODS, and MUP. Other style sheets allow updating early MEI files to more recent versions of the schema which serves as workaround for missing backward compatibility.

Most converters feature limited support for MEI's various encoding styles and features. Here are some typical examples:

- default durations and octavings (specified at `scoreDef`, `staffDef` and/or `layerDef` elements) are often ignored,
- missing fallback strategies for overfull and "underfull" measures,
- missing routines for expansion of repetitions, abbreviations and "slacker" encodings (e.g. elements with a `copyof` attribute),
- no support for nested tuplets, repetitions and expansions,
- no or incomplete support for transposing instruments,
- no consideration of free accidentals that are not explicitly linked to a note,
- no or incomplete support for unorthodox key signatures and accidentals (e.g., quarter tones),
- relevant note and rest information are expected to be encoded explicitly at the note or rest,
- no strategies for competing information on different hierarchical levels (e.g. transpositions on the score level, staff level and locally),
- no support for mixed encodings, i.e. when logical, gestural and visual information complement each other.

This leads to incomplete or wrong data export. With meico we intend to provide a conversion framework that complements existing approaches by a broader and more detailed support of MEI features (specifically regarding the common music notation part of MEI), advanced MIDI support, audio export, and versatile possibilities of integration with applications, other projects and workflows.

2.1 Architecture, Usage & Features

Meico is implemented as Java library and can be obtained from GitHub.[2] It holds fully-fledged classes to interface and process MEI, JSON, MIDI and audio data. We furthermore defined the format Musical Sequence Markup (MSM), an intermediate XML format that is particularly devised as result of MEI preprocessing. As such it is much easier to navigate, analyze and do further conversions. It features some structural similarities with MidiXML but without many of the limitations of MIDI and with several extensions.[3] It is also easy to link back to the MEI source because MSM keeps the original `xml:id` values.

The release file `meico.jar` can be used as programming library, but is also a runnable Jar file that provides two standalone applications. *Commandline mode* is best suited as a service that is accessed via scripted calls. *Window mode* provides a graphical user interface (see figure 1 for the first revision GUI and figure 2 for the second revision) which is made for end users (e.g. music editors). The source

[1] http://www.verovio.org, last access: Apr. 2018.

[2] https://github.com/cemfi/meico, last access: July 2018.

[3] such as floating-point time and pitch values, pitch names, key signatures with arbitrary accidentals, and extended sequencing.

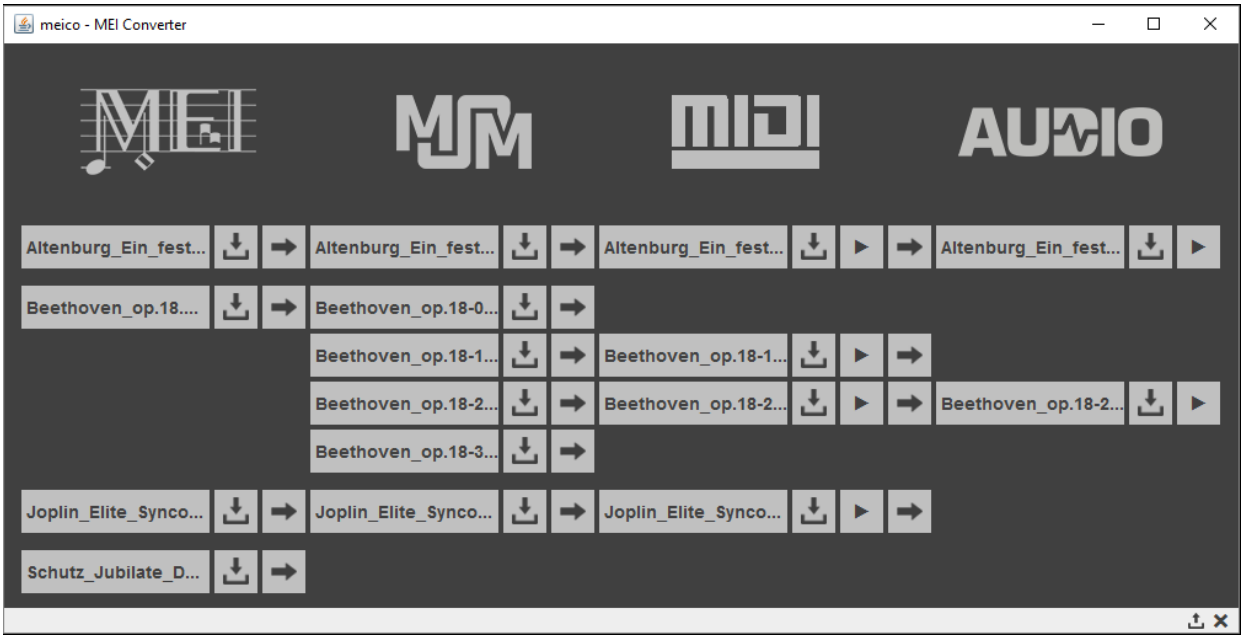

Figure 1: In window mode, meico presents a graphical user interface This is the first revision of meico's GUI.

files of these two modes demonstrate usage of meico as programming library and are extensively commented to serve as tutorial. A code example is given in listing 1. An exemplary Python script, `meicoPy.py`, is also included in the meico sources and provides a tutorial for direct usage of meico in Python. A REST API and docker container make it easy to run meico as a web service. Hence, there are several ways to utilize meico's functionality.

Meico's core functionality comprises:

- reading and writing MEI, MSM, chroma/pitch data (JSON), MIDI, and audio (Wave, MP3) files,
- MEI-to-MSM conversion with editable time resolution (in pulses per quarter) and the option to process `expansion` elements (i.e. sequencing commands in MEI),
- MSM-to pitch data conversion (encoded in JSON) with freely definable reference frequencies and scales, an instance of this is chroma export,
- MSM-to-MIDI conversion with editable basic tempo and the option to expand repetitions as far as they were not covered by `expansion` elements,
- MIDI-to-audio rendering with support for SoundFont (sf2) and Downloadable Sounds (dls),
- an XSLT interface to apply XSL transform style sheets to MEI and MSM documents, e.g. the Music Encoding Initiative's XSLT style sheets [9] to obtain support for MusicXML and a series of further formats.

During MEI-to-MSM conversion, meico automatically computes and sets the minimal required time resolution (in pulses per quarter) so that even the shortest note values can be represented on an integer basis. This is done by finding the shortest note value of the piece, taking dotted note values into consideration and also rest values. If the user specifies a preferred resolution, meico ensures that this

```
import java.io.*;
import meico.*;

public static void main(String[] args) {
  File meiFile = new File("path\\my.mei");
  Mei mei = new Mei(meiFile); // load MEI data
  mei.validate();             // validate input
  mei.addIds();               // add missing IDs

  List<Msm> msms = mei.exportMsm();// MEI to MSM
  // result is a list of MSMs, one per movement

  for (Msm msm : msms) {      // for each MSM
    // expand repetitions
    msm.resolveSequencingMaps();

    // export and store chroma features
    Pitches chr = msm.exportChroma();
    chr.writePitches();       // store JSON file

    // export and store MIDI
    Midi midi = msm.exportMidi();
    midi.writeMidi();         // store MIDI file

    // export and store audio
    Audio audio = midi.exportAudio();
    audio.writeAudio();       // store wav file
    audio.writeMp3();         // store MP3 file
  }

  // MusicXML export via XSLT
  String mxl =
    mei.xslTransformToString("mei2musicxml.xsl");
  Helper.writeStringToFile(mxl, "path\\my.mxl");
}
```

Listing 1: A Java code example for the usage of meico. A minimal conversion example would just read the MEI file and call the export methods.

Figure 2: After several extensions of meico's functionality conversion paths became more branched deeming the first revision graphical user interface inappropriate. The second revision GUI, shown here, presents a graph visualization of conversion paths, more accessible functionality via radial menus, a more advanced audio and MIDI player, and an integrated WebView to access online services such as the Verovio score rendering.

is not below the minimal resolution required to discriminate these values. In MSM, pitch, duration and timing values can be specified in floating-point format. This allows meico to support the whole range of accidentals (e.g. quarter-tone accidentals, micro-tuning) and complex rhythmic situations (e.g. nested tuplets, multi-dotting, ties). Meico keeps track of transpositions, octaving, key signature, accidentals, default values, and resolves any interplay and ambiguity between them. Therefore, it first looks for gestural information (how the music is actually played), than for logical and finally tries to interpret visual information as far as possible. This prioritization is indispensable for the resolution of ambiguities that frequently occur in MEI encodings.

Abbreviation elements in MEI (`beatRpt`, `halfmRpt`, `mRpt`, `mRpt2`, and `multiRpt`) and "slacker" encodings (e.g. elements with a `copyof` attribute) are resolved. Repetition signs and endings in MEI are translated to MSM `sequencingMaps` which can be resolved to repetition-free sequences. Again, complex and unorthodox situations (e.g. nested and overlapping repetitions) are supported.

Meico has a routine to handle overfull and "underfull" measures which ensures synchronization of all musical parts. Names of staffs and staff groups in MEI are kept in MSM; an instrument dictionary and several string matching algorithms determine MIDI program change numbers from these. Of course, this latter feature requires the used sound bank to comply with the General MIDI standard. Alternatively, the dictionary can be adapted or exchanged.

Rather than encoding only one definitive text, an MEI instance can include divergent variants from multiple sources and, hence, enables several readings of the work. These variants are encapsulated in different elements that indicate the character of the variant, such as correction, regularization, substitution, original, unclear, abbreviation etc. While it is generally up to the user to resolve this ambiguity and make the decision for a desired reading meico also provides a default method that prioritizes corrected and complete variants. A tool that supports users preparing their own readings is Sequence Editor.[4]

The generated MIDI and audio can be written to the file system or played back directly with meico's built-in players which proved

[4] nashira.uni-paderborn.de:5555, last access: July 2018.

useful as debugging feature for music editors during encoding. Encoding errors are often easier to hear than to see. Further support features for music editors include:

- MEI validation against MEI Common Music Notation Schema (`mei-CMN.rng`),
- `xml:id` generation for a series of MEI elements,
- transformation of `tie` elements to `tie` attributes,
- expansion of elements with the "slacker" attribute `copyof`; these elements are replaced by deep copies of the referred element, incl. the generation of unique `xml:id`,
- serialization of `expansion`s to generate "through-composed" MEI code.

The resulting verbose MEI document can then be added to a digital music edition and used for further processing and analyses.

Generally, the interpretation of MEI encodings is not fully standardized. Therefore it is necessary for meico to be accompanied by a rather extensive documentation of supported MEI elements and attributes and how they are interpreted.

2.2 Limitations & Future Plans

In comparison with other conversion frameworks, particularly regarding MEI-to-MIDI/audio conversion, meico currently implements the broadest and most detailed coverage of MEI common music notation features and stands up to most situations that typically occur in MEI encodings. It is, nonetheless, far from a complete MEI coverage (as currently all MEI applications due to the format's complexity). Meico focusses on those aspects of MEI that are relevant for proper and unambiguous music playback and analyses. This is a distinct difference to, for instance, Verovio that is much better devised for music typesetting, a scenario that meico does not address, so far. Meico provides access to Verovio's capabilities via an integrated WebView.

As the conversion paths became more branched during meico's further development we saw the demand for a different user interface approach. Hence, replaced the relatively limited first revision GUI (figure 1) by the more versatile second revision (figure 2) that creates a graph visualization of the conversion paths. Its current graph layouting algorithm is relatively static, dragging one node does not affect others connected to it. This makes the manual arrangement of the data cumbersome. Thus, we consider adding a mass-spring system-like behavior in future revisions.

We plan to add further import and export formats such as MusicXML and Humdrum. In addition to this, we also plan to expand the MEI coverage. So far, meico skips all elements that address musical performance aspects, such as articulation, dynamics, tempo, and ornamentation signs (e.g. trills, glissandi, tremoli). We deliberately made this decision to keep a clear separation of the musical "raw material" and its performance. A future expansion will keep this separation and export performance data to a specialized format called Music Performance Markup (MPM) [3] which is then used to render expressive MIDI sequences, such as described in [2]. At that point, meico can become part of electronic music production workflows and interfaces parameters of expressive performance far beyond standard humanizing.

3 SCORETUBE: AUDIO-SCORE ALIGNMENT

This section presents a typical application of meico, an audio-to-score alignment tool that is used in the context of digital music editions. Music editions are traditionally published as printed scores representing the editor's definitive text. The presentation format of digital music editions, on the other hand, is not limited to a printed score [6]. A typical digital music edition comprises the score data, the digitized source materials (autographs, manuscripts, prints), allows to switch between them or display them synchronously, and indicates differences between them, including a critical discussion. All this can be accompanied by audio recordings. If the music was previously never recorded, meico may be used to generate sound examples. In the following, however, we will assume that human recordings exist and should become part of the edition.

To comply with the overall idea of presenting the different materials and media in a synchronized way, so that the edition becomes more easily accessible, the audio material has to be aligned with the score and both are presented at once. The *FreiDi:syncPlayer*[5] is an instance of this idea. It is based on timestamps that were manually created to overcome the difficulties in processing MEI data [14]. This is where meico comes into play. Our tool, ScoreTube, provides automatic alignment of MEI score data with audio and video recordings.[6] ScoreTube provides synchronized display of MEI score and audio/video in web browsers and allows navigation through these by clicking at the desired playback position in the rendered score or the media player, as shown in figure 3.

ScoreTube's alignment algorithm utilizes a classic Dynamic Time Warping (DTW) approach based on chroma features [11] that is implemented in the *librosa* library [12]. Chroma features are obtained from both audio and MEI data. While chroma features can easily be extracted from audio files via established tools such as librosa, it is less convenient for MEI scores, so far. This task is now conveniently solved by meico. The result of the alignment algorithm is a dictionary that maps `xml:id` values from MEI score to time positions in the audio file.

Even though librosa's DTW implementation is relatively basic, it performed already considerably well in our tests. It can, nonetheless, be replaced by alternative alignment algorithms that are frequently released in the Music Information Retrieval community, see, for instance, Thomas et al.'s [16] state-of-the-art report. We recently added a novel algorithm, Transposition-Aware Dynamic Time Warping (TA-DTW) that is also able to compute alignments in the presence of pitch offsets and pitch drifts, which is particularly relevant when dealing with early music and a cappella music. A detailed description and evaluation of TA-DTW is given in [20].

Score rendering in ScoreTube is done via the Verovio music typesetting framework [15]. It outputs the typeset MEI score in SVG format and—just as meico—retains the original MEI `xml:id` values of notes, rests, measures etc. which is common practice in the MEI ecosystem. Having now both, the IDs in the SVG and in the alignment, any coordinate in the SVG is mapped to a time position in the audio file. Relative positions between two notes are mapped to relative positions between the corresponding alignment timestamps. With this, it is possible to draw a continuous playback cursor (red

[5] freischuetz-digital.de/demos/syncPlayer/, last access: July 2018.
[6] https://zenmem.de/score-align/, last access: July 2018.

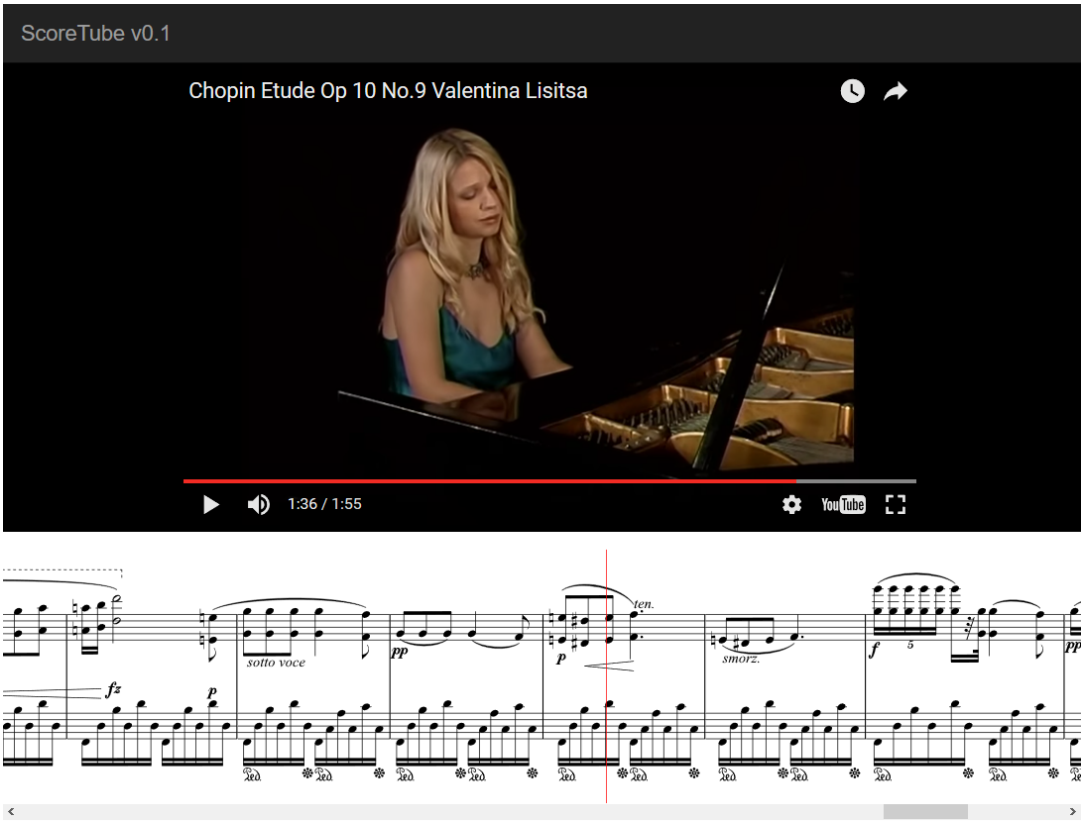

Figure 3: ScoreTube is an Audio-to-Score Alignment solution for MEI data.

vertical line in figure 3) and select exact playback positions by clicking in the score.

The ScoreTube score follower has already been used successfully in assistive music production tools [17–19] and as part of presentation software for digital music editions to seamlessly switch between various recordings of an edited work. Further applications range from score-assisted media players, e.g. in an educational context, to complex musical interfaces incorporating an interactive score view. Further developments will focus on incorporating alignment for scanned sheet music such as autographs and facsimiles.

4 SUMMARY

Digital music editions and the corresponding representation format MEI are worth being considered more frequently in other music technology-related communities such as Music Information Retrieval and music production. With the MEI converter framework meico, we hope to make access, processing and usage of MEI encoded music more convenient and appealing. It allows to transform MEI data into formats that are more common in other application contexts. ScoreTube demonstrates this with an audio-to-score alignment tool on the basis of MEI scores.

Meico became already part of the workflow of music editors and is in use in several digital music edition projects. It complements Verovio's score rendering with quickly generated sound examples that make it a lot easier to detect encoding errors. These sound examples can also be added to the final music edition which is particularly relevant when no other recordings of the music exist. Meico provides a first impression of how the music will sound.

Acknowledgements This work is part of the project "Center Music – Edition – Media", funded by the German Federal Ministry of Education and Research (BMBF 01UG1714A–B).

REFERENCES

[1] C. Antila. 2017. MEI module for music21. https://github.com/cuthbertLab/music21/tree/master/music21/mei. last access: July 2018.
[2] A. Berndt. 2015. Formalizing Expressive Music Performance Phenomena. In *Works in Audio and Music Technology*, A. Berndt (Ed.). TUDpress, Dresden, Germany, Chapter 4, 97–128.
[3] A. Berndt and B. W. Bohl. 2018. Formale Beschreibung musikalischer Interpretationen. In *Aufführung und Edition 2018, 17. Int. Tagung der Arbeitsgemeinschaft für germanistische Edition*. Goethe-Universität Frankfurt am Main, Frankfurt am Main, Germany.
[4] P. J. Billam. 2007. mei2mid—Perl script to convert MEI to MIDI. http://www.pjb.com.au/midi/mei2mid.html. last access: July 2018.
[5] S. Cox, M. Hartwig, and R. Sänger. 2015. Beethovens Werkstatt: Genetische Textkritik und Digitale Musikedition. *Forum Musikbibliothek* 36, 2 (July 2015), 13–20.
[6] M.-J.-B. Favart and A. B. Blaise. 2016. *Annette et Lubin: Comédie en un acte en vers, mêlée d'ariettes et de vaudevilles* (historical-critical hybrid ed.). OPERA: Spektrum des europäischen Musiktheaters in Einzeleditionen, Vol. 2. Bärenreiter, Akademie der Wissenschaften und der Literatur, Mainz, Kassel, Germany. A. Münzmay and J. Droese (eds.).
[7] A. Hadjakos, A. Berndt, and S. Waloschek. 2015. Synchronizing Spatially Distributed Musical Ensembles. In *12th Sound and Music Computing Conf. (SMC15)*. Maynooth University, Maynooth, Ireland.

[8] A. Hankinson, Roland P., and I. Fujinaga. 2011. The Music Encoding Initiative as a Document-Encoding Framework. In *Int. Society for Music Information Retrieval Conf. (ISMIR)*. Int. Society for Music Information Retrieval, Miami, Florida, USA, 293–298.

[9] A. Hankinson and P. Roland. 2016. Tools for working with or transforming MEI Encodings. https://github.com/music-encoding/encoding-tools. last access: July 2018.

[10] A. Hankinson and T. Weber. 2018. Sibelius to MEI Plugin. https://github.com/music-encoding/sibmei. last access: July 2018.

[11] N. Hu, R. B. Dannenberg, and G. Tzanetakis. 2003. Polyphonic audio matching and alignment for music retrieval. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on*. IEEE, 185–188.

[12] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, D. Ellis, F.-R. Stoter, D. Repetto, S. Waloschek, C. J. Carr, S. Kranzler, K. Choi, P. Viktorin, J. F. Santos, Adrian Holovaty, W. Pimenta, and H. Lee. 2017. librosa 0.5.0. https://doi.org/10.5281/zenodo.293021 last access: July 2018.

[13] Music Encoding Initiative. 2017. An Introduction to MEI. http://music-encoding.org/resources/introduction.html. last access: July 2018.

[14] T. Prätzlich. 2016. *Freischütz Digital: Processing Audio Signals in Complex Music Scenarios*. Ph.D. Dissertation. Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany.

[15] L. Pugin, R. Zitellini, and P. Roland. 2014. Verovio: A Library For Engraving MEI Music Notation Into SVG. In *Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR)*. Int. Society for Music Information Retrieval, Taipei, Taiwan.

[16] V. Thomas, C. Fremerey, M. Müller, and M. Clausen. 2012. Linking Sheet Music and Audio – Challenges and New Approaches. In *Dagstuhl Follow-Ups*, M. Müller, M. Goto, and M. Schedl (Eds.), Vol. 3. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 1–22.

[17] S. Waloschek. 2017. *Interaktive Partituren in der Musikproduktion*. Master's thesis. Ostwestfalen-Lippe University of Applied Sciences, Center of Music and Film Informatics, Detmold, Germany.

[18] S. Waloschek, A. Berndt, B. W. Bohl, and A. Hadjakos. 2016. Accelerating the Editing Phase in Music Productions using Interactive Scores. In *Proc. of the 2nd AES Workshop on Intelligent Music Production*. Queen Mary University of London, Center for Digital Music, Audio Engineering Society (AES), London, UK.

[19] S. Waloschek, A. Berndt, B. W. Bohl, and A. Hadjakos. 2016. Interactive Scores in Classical Music Production. In *Proc. of the 17th Int. Society for Music Information Retrieval Conf. (ISMIR) 2016*, M. Mandel (Ed.). New York University, Int. Society for Music Information Retrieval, New York, NY, USA.

[20] S. Waloschek and A. Hadjakos. 2018. Driftin' down the scale: Dynamic time warping in the presence of pitch drift and transpositions. In *Proc. of the 19th Int. Society for Music Information Retrieval Conf. (ISMIR)*. Int. Society for Music Information Retrieval, Paris, France.