MeRIT: An interactive annotation tool for mensural rhythms

Anna Plaksin
aplaksin@uni-mainz.de
IKM Abt. Musikwissenschaft, Johannes Gutenberg-Universität Mainz
Mainz, Germany

David Lewis
david.lewis@oerc.ox.ac.uk
Oxford e-Research Centre, University of Oxford
Oxford, UK

ABSTRACT

We introduce MeRIT, the Mensural Rhythm Interpretation Tool, a client-side JavaScript tool that interprets the rhythmic notation of pre-modern polyphonic music using rules derived from contemporary theory. The interpretation derived from the tool is written in standards-compliant MEI, including details of the rules being applied at each point and a fine-grained metrical analysis.

We discuss the importance of system modularity and different levels of evaluation and describe a user feedback system that we have implemented, supporting corrections, detailed introspection and evaluation and also pedagogical use, training musicians to read the notation. This user interaction is facilitated by a system architecture that provides an API encapsulation of the MEI interaction, giving annotations provided by the user and interpreter a common interface.

CCS CONCEPTS

• **Applied computing → Sound and music computing**.

ACM Reference Format:
Anna Plaksin and David Lewis. 2022. MeRIT: An interactive annotation tool for mensural rhythms. In *9th International Conference on Digital Libraries for Musicology (DLfM2022), July 28, 2022, Prague, Czech Republic.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3543882.3543888

1 INTRODUCTION

We often build tools in digital musicology to facilitate (editorial) work or to make larger corpora accessible, but in the process of development, it also becomes clear that the requisite conceptualisation and modelling also opens up a different perspective that goes beyond their status as mere workflow components. Tool development can also take a less deterministic approach – e.g. by trying to emulate cognitive processes described in music theory – and here we move into an area where much exploration is still necessary and possible. In this context, the development work requires a reflective component, with evaluation and introspection at its core. Thus, the question must always be asked: What can a tool be used for in the first place? Does it fulfil the task for which it was built? And can it fulfil this task satisfactorily at all? Under what conditions? At the same time, these questions open up new perspectives and foster new developments. This potential will be discussed here reporting on MeRIT, the Mensural Rhythm Interpretation Tool.[1]

MeRIT is a tool that interprets mensural rhythm. It provides an interface that supports full automation and live evaluation for scoring-up of sources, but also tools for teaching the notation. In this paper, we show how the development and evaluation of applications can go hand in hand and thus open up further application scenarios.

2 ON THE INTERPRETATION OF MUSICAL TIME

The idea for MeRIT follows on directly from the digital edition of Johannes Tinctoris' music-theoretical writings. It addresses a central issue confronting those who work with sources of pre-modern polyphonic music, whether as editors, analysts, or performers. Mensural notation is the most common system of music notation for polyphonic vocal music from the second half of the 13th century to c. 1600. Like modern western notation it contains rhythmic information throughout, but its notation is highly contextual. Unlike modern notation, in which all notes are divisible only by two (with well-defined exceptions called 'tuplets') and are therefore definite in duration, many notes in mensural notation could be divided either by two or by three without visual distinction. Musical context can also impose further changes to the note's duration. These manipulations are widely known as imperfection and alteration – the former describes the contextual shortening of triple durations, the latter the doubling of duple durations.

The main goal was to develop automated approaches to the interpretation of durations in mensural notation from the perspective of the 15th-century music theorist Johannes Tinctoris (c. 1435–1511).[2] Creating a system resting upon the writings of one particular theorist, our intention is not solely focused on the results but follows an introspective approach – this sets it apart from other efforts like the *scoring-up tool* developed by Martha Thomae[Thomae 2017], that is part of the Measuring Polyphony Editor[Desmond 2020]. Through creating our software, we hope to learn more about music theory and gain further insights into Tinctoris' relationship with the wider practice of his time. Our tools should not be limited to analytical uses, but also to help users learn how to interpret the mensural understanding and notation of rhythm.

From a more general perspective of software development, the task is to create a system that conducts the automated annotation of musical durations through the application of rules. Along with this, the introspective approach generates the demand for a high

DLfM2022, July 28, 2022, Prague, Czech Republic

ACM ISBN 978-1-4503-9668-4/22/07…$15.00
https://doi.org/10.1145/3543882.3543888

[1] The tool is available for testing at https://interpreter.earlymusictheory.org/.
[2] For a biographical outline, see https://earlymusictheory.org/Tinctoris/Tinctoris/BiographicalOutline.

transparency of decision-making – the software system needs to communicate its processing to the user. In addition, one key concern in the development was a high level of flexiblitiy with regard to different user scenarios.

3 SYSTEMATIC EVALUATION THROUGH USER FEEDBACK

The user-centred approach that is implicit in this scenario goes beyond the detailed and comprehensible presentation of results. As a tool that is built to explore the application of a theoretical rule system to music, a user scenario does not end when the interpreter completes its analysis – it extends into reviewing the results under various conditions, and being able to generate reliable statements on the quality of the annotations in real-world examples. Consequently, such a system must provide functionality for testing the consistency and transferability of the implemented rule system by conducting a systematic evaluation.

This is a central requirement that needs to be addressed throughout the whole development process. From a theoretical perspective the evaluation may be structured into three broad levels:

Integrity of the implementation Does the implementation do what was intended?

Functionality of the modelling Has the model been satisfactorily formalised?

Transferability of the modelling Do other examples (works by other composers, other sources) follow the modelled rule system at all?

On a basic level, we must ensure how well the implementation matches the modelling. Ensuring the integrity of the implemented system is important not only in terms of software quality, but it also enables any further reasoning described in the following steps. The two following steps, however, are inconceivable without a critical engagement with Tinctoris' writings. While the aspect of functionality points directly at the tension between music theory, its understanding, and its feasibility, the question of transferability aims at the relationship between Tinctoris and the wider practice. Regarding stage two, one key finding in the development process was, that the immediate implementation of Tinctoris' writings could not be achieved since they do not contain rules to apply, but rather for classifying and understanding alteration and imperfection.

Overall, these levels add up to an iterative process of reasoning in itself, including the selection of examples for various purposes – e.g. curated artificial test cases to ensure expected behaviour vs. real-world examples –, the inspection of results and the compilation of differences. Central to this is the potential of recording targeted user feedback and comparing it with the results.

4 MERIT AS A MODULAR SYSTEM

The interpreter logic has been embedded in a modular system, implemented in client-side JavaScript which places the user feedback on an equal footing with the results of the interpreter itself. It has been realised as a client-side only web application working with encodings following the MEI Guidelines for mensural notation. The following sections introduce the components of the system, their requirements, and the assumptions on which they are based.

4.1 The interpreter logic as component

The interpreter logic itself conducts multiple iterative steps to analyse the mensural framework and apply rules of imperfection and alteration. Before this can take place, the prevailing rhythmic structure – the mensuration – must be known, and dot symbols must be roughly classified. The initial step is the detection of mensurally uniform blocks. After these have been determined, the analysis only takes place within such a block.

The analysis itself comprises three different steps – the resolution of beat independent cases, the calculation of the overall rhythmic framework of a block, and the analysis of context-dependent cases. Beat-independent cases can be resolved without any prior knowledge about a note's position. The include, *inter alia*, the resolution of rests, regularly imperfect, unalterable notes and instances of what is called *similis ante similem perfecta*[Apel 1961, p. 108]. Based on this, a preliminary calculation of the mensural framework and the rhythmical position of each note or rest within can be performed. Then, alternating steps of gradually resolving the context-dependant cases, where a rough estimation of a note's position is necessary, and re-calculating the mensural framework are performed until no further progress can be made (see figure 1).

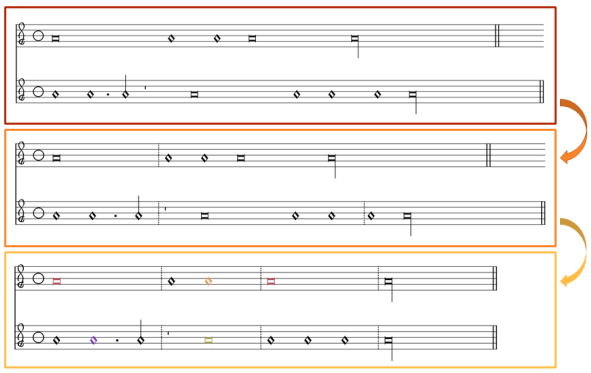

Figure 1: Steps of interpretation: 1. unresolved example, 2. analysis of beat independent rules, 3. fully resolved example including indicative colours and breve boundaries.

Every procedure results in the production of specific markup. In addition to the usual markup for MEI to describe mensural rhythm, the interpreter also adds a reference to the rule that was applied in the particular case. The markup describing the mensural framework is not only essential for all calculations, but also specific to the interpreter. It contains the duration of a specific event in numbers of minims[3] and the starting position of an event in minims as well as described in units of note levels.

4.2 User interface and overall processing

For the development of the user feedback functionality, the following sequence of steps can be assumed:

(1) Load MEI file
(2) Interpreter annotates file (incl. storing its changes)

[3]The minim is a note type in mensural notation that is the smallest type involved in diverging division ratios. Smaller types are always bipartite.

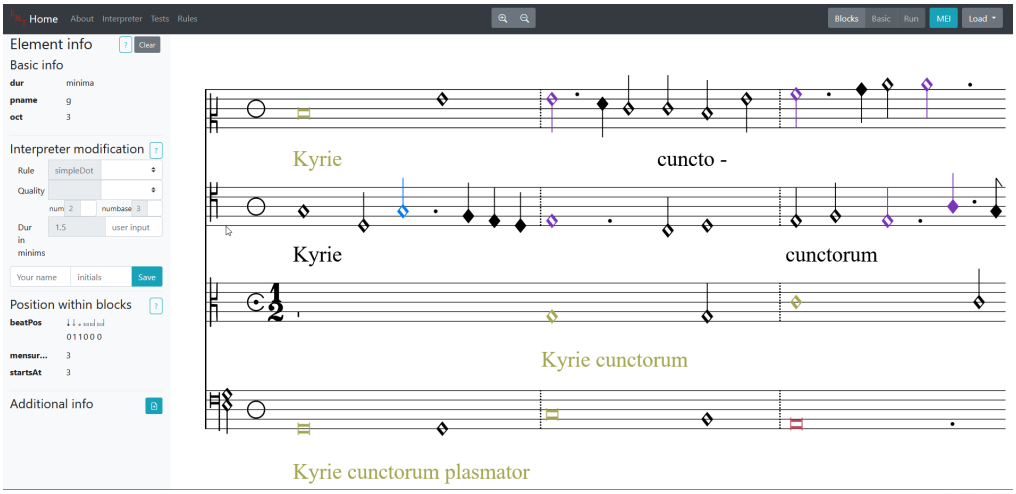

Figure 2: The MeRIT user interface.

(3) User corrects annotations (with their corrections stored)

We make no functional distinction between the interaction of a user or the interpreter with the musical document. The system rather serves as an environment for successive annotations by two independent agents. It manages the interaction with the MEI document and shares its content with both agents. The calculation of the mensural framework may be seen as an exception to this, but since it is based completely on the duration in minims, it turned out to be more feasible to keep it read-only and let the user interact only with its input. It also allows the rendering to be updated based on user input.

The user operates through a graphical user interface. The main objective in the design was to put the content in the foreground and to allow the user as much overview as possible of the musical context. In order not to cover this with elements, the essential components were positioned in a sidebar. The interface allows inspection of the results on two levels. Interpretative markup is indicated by the colouring of the objects in the score. If this level of detail is not sufficient, the user can click on an object in the score to inspect the detailed results of the interpreter and also enter his or her corrections.

4.3 Capturing diverging opinions

The interpreter does not only generate standard MEI markup but also specific mark-up that is stored in a standards-compliant way. Instead of making use of a specialised customisation, an encoding combining regular standard MEI attributes and separate annotation has been preferred. The standard attributes are necessary for the correct interpretation of the file, e.g. rendering with Verovio.[Pugin et al. 2014] Annotations are realised in MEI with the <annot> element. It “provides a statement explaining the text or indicating the basis for an assertion" and "can be used for both general comments and for annotations of the musical text”.[Music Encoding Initiative 2019c] As a control event, it makes "no independent contribution to that flow of music" but provides information about other referenced elements.[Music Encoding Initiative 2019b] In addition, the

```
<note dur="semibrevis" pname="a" oct="4" xml:id="someNote"/>
```

```
<note dur="semibrevis" pname="a" oct="4" xml:id="someNote" dur.metrical="4"
dur.quality="altera"/>
<!-- [...] -->
<annot startid="#someNote" type="mensural-interpreter" audience="private">
    <annot type="startsAt" resp="#mensural-interpreter">4</annot>
    <annot type="beatPos" resp="#mensural-interpreter">0, 0, 2, 0, 0, 0</annot>
    <annot type="dur.metrical" resp="#mensural-interpreter">4</annot>
    <annot type="rule" resp="#mensural-interpreter">A.2</annot>
    <annot type="dur.quality" resp="#mensural-interpreter">altera</annot>
</annot>
```

```
<annot type="rule" resp="#mensural-interpreter">
    <choice>
        <sic resp="#mensural-interpreter">A.2</sic>
        <corr resp="#ap">unalteredImperfectAfter</corr>
    </choice>
</annot>
```

Figure 3: Encoding of annotations and diverging opinions: The data input on the top receives standard attributes and annotated property during the interpretation (middle). The user feedback is stored inside these annotations as editorial markup (bottom).

<annot> element offers great flexibility – it may contain a variety of elements including itself and offers a lot of attributes especially for referencing other parts of the document and analytical purposes. This made it easy to map the interaction at the data level. Every object in the score receives an annotation in the course of processing that contains sub-annotations for every property. For the purpose of interpretation only, it would be preferable to store only custom properties within annotations, but the redundant information facilitates our user feedback functionality.

Gathering the results of the interpreter and accompanying user feedback can be seen as inherently about recording disagreement. In the particular case of a user marking and correcting errors, these diverging opinions can be explicitly treated as such and be encoded following MEI practice using the standard’s <sic> and <corr> within a <choice> element, including an indication of the responsible agent.[Music Encoding Initiative 2019a] The use case of correcting also implies the order of the markup: the annotation value

of the first agent is stored within <sic> while the value added by the second agent is stored as <corr> (see figure 3). Key information about the responsible agents involved in the annotation are stored within the document header as application info and responsibility statement. In addition, the stages of interaction are recorded in the revision description.

4.4 Facilitating interactivity through abstraction

Since it does not matter, from a software point of view, which agent is reading or writing information at any given time, it is practical to bundle all interaction with the document into one interface. For that reason, the system introduces an *I/O Handler* as an abstraction layer that manages the interaction with the MEI document. This component fully realises the conceptional abstraction of the interpreter and the user acting as agents annotating an MEI document (see fig. 4).

By aggregating every interaction with the MEI document, it ensures that the markup is complete and consistent. In the process, the handling of the quite complex markup, which consists of attributes on the one hand and annotations on the other hand, is considerably simplified. For this purpose, it completely encapsulates the MEI annotation, allowing it to be manipulated programmatically, without direct editing of the XML and so managing the markup internally. It controls both the content of the annotations including the editorial markup and which properties are to be stored as attributes. Externally, the *I/O Handler* reads or returns the content of the annotations as JavaScript objects or as text string, if only values of a certain state or agent are to be read. At the same time, the interpreter itself becomes largely format-agnostic and it is easier to make subsequent adjustments to the markup or change the format or method of provision at all.

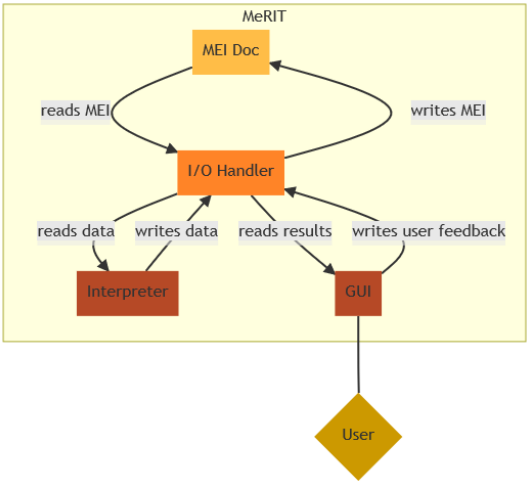

Figure 4: MeRIT's modular architecture: Both the automated Interpreter component and the user-facing GUI manipulate the MEI via an *I/O Handler* that presents functionality for interacting with the encoding to streamline the annotation process.

5 CONCLUSION AND FUTURE WORK

We have shown how the MeRIT application is not only a tool for the interpretation of mensural rhythms, but also addresses the need for systematic evaluation that arises during the development of automated annotation or analytical routines in general. The aim of fostering systematic evaluation through an interactive user feedback function has led to a modular system that opens up wider perspectives. On the one hand, pursuing a modular architecture allows a compromise between unique use cases with specific requirements and essential sustainability and reusability. By identifying generic workflow components, encapsulating and (re)combining them, they can be made available for reuse instead of implementing them over and over again. On the other hand, the presented approach is not limited to evaluation scenarios alone but supports interactive use cases in general, as the sequence, granularity or number of the interactions or involved agents is not prescribed: the collection of user feedback is therefore only one case of many. Although the process is currently fully automatic, our more granular approach could easily support a semi-automated scenario of assisted annotation. As well the pursued learning resource can be realised based on turning around the order of annotating agents. Receiving user feedback first followed by the interpreter output and comparing both, also makes it possible to render live suggestions.

The present system is currently a working prototype implementation. Future development will both focus on evaluating the transferability of the rule set and explore the use of MeRIT in learning scenarios and the potential for adding further thematic modules. To ensure the potential reusability in other scenarios, efforts need to be put into the consolidation of the presented components, particularly the document interaction layer and the *I/O Handler* as annotation management interface.

ACKNOWLEDGMENTS

MeRIT was developed by the UK Arts and Humanities Research Council (AHRC) funded project "Interpreting the Mensural Notation: An Expert System Based on the Theory of Johannes Tinctoris" (AH/P013910/1). It builds on the project "The Complete Theoretical Works of Johannes Tinctoris: A New Digital Edition" (AH/I003827/1).

We thank our research collaborators and investigators, Ronald Woodley, Jeffrey J Dean, Christian Goursauld and Adam Whittaker, for their contributions to the theoretical underpinning of our software and user interface design, and the Music Encoding Initiative's Mensural Notation Special Interest Group for support in development of the mensural notation module of MEI on which our work depends.

REFERENCES

Willi Apel. 1961. *The Notation of Polyphonic Music, 900-1600* (5th ed. rev. and with commentary ed.). Number 38 in The Mediaeval Academy of America. Publication. Mediaeval Academy of America, Cambridge, Mass. https://archive.org/details/notationofpolyph00apel/

Karen Desmond. 2020. Next Steps for Measuring Polyphony: A Prototype Editor for Encoding Mensural Music. In *Music Encoding Conference Proceedings 26-29 May, 2020 Tufts University, Boston (USA)*, Elsa de Luca and Julia Flanders (Eds.). Humanities Commons, 121–124. https://doi.org/10.17613/5K88-9Z02

Music Encoding Initiative. 2019a. *11.2. Editorial Markup.* Vol. (Version 4.0.1). https://music-encoding.org/guidelines/v4/content/scholarlyediting.html#editTrans

Music Encoding Initiative. 2019b. *2. Shared Concepts in MEI.* Vol. (Version 4.0.1). https://music-encoding.org/guidelines/v4/content/shared.html

Music Encoding Initiative. 2019c. *[Element definition:] <annot>*. Vol. (Version 4.0.1). https://music-encoding.org/guidelines/v4/elements/annot.html

Laurent Pugin, Rodolfo Zitellini, and Perry Roland. 2014. Verovio. A Library for Engraving MEI Music Notation into SVG. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, Hsin-Min Wang, Yi-Hsuan Yang, and Jin Ha Lee (Eds.). International Society for Music Information Retrieval, 107–112. http://www.terasoft.com.tw/conf/ismir2014/proceedings/T020_221_Paper.pdf

Martha E. Thomae. 2017. *Automatic scoring up of mensural music using perfect mensurations, 1300-1550.* Master's Thesis. McGill University, Montreal. https://escholarship.mcgill.ca/concern/theses/tm70mz01z