SEPARATING PRESENTATION AND CONTENT IN MEI

Laurent Pugin
Swiss RISM / Fribourg University
laurent.pugin@rism-ch.org

Johannes Kepper
Edirom
kepper@edirom.de

Perry Roland
University of Virginia
pdr4h@eservices.virginia.edu

Maja Hartwig
Edirom
maja.hartwig@gmx.de

Andrew Hankinson
McGill University, Schulich School of Music
andrew.hankinson@mail.mcgill.ca

ABSTRACT

Common Western music notation is traditionally organized on staves that can be grouped into systems. When multiple systems appear on a page, they are arranged from the top to the bottom of the page, similar to lines of words in a text document. Encoding music notation documents for printing requires this arrangement to be captured. However, in the music notation model proposed by the Music Encoding Initiative (MEI), the hierarchy of the XML sub-tree representing the music emphasizes the content rather than the layout. Since systems and pages do not coincide with the musical content, they are encoded in a secondary hierarchy that contains very limited information. In this paper, we present a complementary solution for augmenting the level of detail of the layout of musical documents; that is, the layout information can be encoded in a separate sub-tree with cross-references to other elements holding the musical content. The major advantage of the proposed solution is that it enables multiple layout descriptions, each describing a different visual instantiation of the same musical content.

1. INTRODUCTION

Common Western music notation is a system made up of structured symbols organized upon a group of horizontal lines, commonly called a "staff", which acts as a bi-dimensional reference system. The horizontal axis represents time while the vertical axis indicates pitch. Staves can be grouped into systems, where the systems contain concurrent streams of musical events aligned vertically and where each staff encompasses a defined pitch range. Systems are arranged across as many pages as necessary to accommodate the musical content. When multiple systems appear on a page, multiple systems are arranged from the top of the page to the bottom, similar to paragraphs in a text document.

Numerous schemes have been developed for encoding music notation [8]. Over the last decade, XML has been increasingly used for defining encoding schemes, for example, in the MusicXML[1] interchange format [2] and the IEEE1599[2] standard [6]. More recently, with a major release in 2010 and with the upcoming 2012 release, the music notation model proposed by the Music Encoding Initiative[3] (MEI) has begun to take a leading role. Developed by a community of scholars, it acts as an extensible music document encoding framework that can be customized for specific needs [5].

For XML encoding schemes, such as MEI, that aim to take into account the graphical context of the notation, the organization of the notation into staves, systems, and pages often needs to be captured. Whereas a page-based approach will have the page at the top of the XML hierarchy, a content-based approach will place an element with semantic meaning at the top of the hierarchy, relegating the visual appearance to a secondary role. Music notation itself is obviously multi-hierarchical, and both approaches reflect valid perspectives. However, a basic principle of XML design is that it requires a single hierarchy to become the primary ordering mechanism of the music notation description. Other hierarchies inherent in music notation may then be implemented using alternative techniques such as standoff markup.

Currently, MEI emphasizes the logical content of the notation. For example, in the case of CMN, it employs measures at the top of the hierarchy. Pages and systems are captured using the same milestone technique that TEI offers; that is, page and system breaks are represented by the empty elements <pb/> and <sb/> respectively. It is fairly easy to convert between measure-based and page-based hierarchies using XSLT stylesheets, analogous to MusicXML's conversion between time-based and part-based file organization. However, there are additional complicating factors in the case of MEI. For example, when multiple sources are described within a single encoding, which is a significant design goal of MEI, the sources do not necessarily agree with regard to page and system breaks. Furthermore, they might use a different

[1] <http://www.makemusic.com/musicxml>
[2] <http://www.mx.dico.unimi.it>
[3] <http://www.music-encoding.org >

score order or map instruments to staves differently. Even the number of instruments or staves may differ between multiple sources. Although MEI is currently capable of dealing with these circumstances, the markup is often verbose, repetitive, and difficult to comprehend quickly. In this paper, we present a complementary module for MEI that provides for more detailed capture of layout information and better separation of musical content and visual presentation. The next section describes the objectives pursued, followed by a section on related work. We then present the module we developed for MEI and conclude the paper with remarks on future work.

2. OBJECTIVES

There are at least two use-cases that would benefit from a clearer separation of layout-related information and the musical content as proposed in this paper. The first use-case is when precise descriptions in the encoding of existing source materials are required. A typical example is the use of MEI as an output of and archival format for optical music recognition (OMR) software applications [4]. In such a use, it is necessary to be able to record the exact position of the elements on the page. In OMR transcriptions, each note, each music symbol, but also each staff and each system requires its coordinate to be stored in the MEI encoding. Diplomatic transcriptions with exact coordinates are not only useful as interchange and training data for adaptive OMR software applications, but they can also be used in digital edition environments for producing transcription image overlays. A diplomatic transcription can be shown directly on top of the original source, either for highlighting a particular aspect of the source or simply for facilitating its readability. Examples already exist for text editions [7], and a similar approach for music could very well be envisaged with MEI.

Such a model would also serve the second use-case, which is the preparation of different renditions from the same musical material, the typical case being an edition of the full score and, in parallel, an edition of the performers' parts. While it is relatively easy to extract parts from a score encoded in MEI, there will always be cases where human intervention will be required to finalize the layout of the parts, whatever the automatic layout capabilities of the rendering software application used. The modifications can include additional dynamic markings, lyrics, directives and similar musical information encoded in nearby staves. The ideal solution is to encode only the layout modifications applied to the parts so that additional changes to the score would automatically be reflected in the parts. This means that a <note> element for which only the stem direction is changed in the layout need not be duplicated. Features like this already exist in some music notation software applications, such as in Sibelius©, which includes a so-called Dynamic Parts™ functionality. However, they are not designed to handle multiple sources. Having an option to record this type of layout information in an optimized manner would certainly be valuable.

2.1 Requirements

In order optimally to increase the level of detail of the documents encoded in MEI, it is necessary to achieve a solution that will not overload the logical sub-tree that holds the musical content. When mingled with notation content, page and system milestone markers complicate the encoding of content. Adding more detailed layout information, such as page size, results in further complication.

The solution should avoid overlapping hierarchy problems whenever possible. Page breaks and system breaks embedded in the content sub-tree represent a non-concurrent hierarchy. Multiple sources requiring different presentation exacerbate the problem by creating multiple instances of non-concurrent hierarchies.

Furthermore, it is important for the solution to limit strictly the amount of duplicated data in the encoding. For example, in the case of "score and parts" editions, when the data for the parts duplicates that of the score, the score data and the parts data may become desynchronized. However, since the duration of a note in the parts should be the same as in the score, employing a reference system eliminates this possibility.

The proposed solution should not require the user to choose between content-based or page-based approaches but should supplement the current content-focused representation of MEI instead. With that in mind it becomes clear that the use of this additional layout information has to be optional – for users, but also for applications. This means that applications unaware of this proposal may safely ignore it, that the additional information provided by this proposal must leave the musical content sub-tree untouched as much as possible, and that links added between the content and the layout elements must not preclude the encoding and decoding of the musical content on its own.

3. RELATED WORK

For some aspects, the problem described above is similar to what is achieved by OMR software applications such as Photoscore© that extend MusicXML in order to store exact positioning information. However, it is done in a non standard way and is application dependent. There are also several standard existing encoding strategies and formats that seem to be relevant to the problem described above. The following section will introduce them briefly and discuss their applicability for describing multiple renditions or sources of the same musical content.

3.1 TEI Encoding Model for Genetic Editions

The aim of providing a detailed model of how content is laid out on the page is similar to the goal pursued by the TEI Workgroup on Genetic Editions [3]. Their model essentially follows a document-centric approach, as opposed to the traditional text-centric approach for TEI. The alternative hierarchy of this model privileges the document and is organized as follows:

- Document
    - Writing surface (page, double page, folium, etc.)
        - Zone
            - Text, lines or tables

The model is designed for encoding complex cases of manuscripts in various stages of creation. Its purpose is to trace and encode their genesis. In that regard, it is different from what we hope to achieve for MEI because this model aims principally for a chronological ordering of zones in one document rather than transcribing or defining the layout of multiple documents separately.

Furthermore, the TEI encoding model for genetic editions is an alternative model for encoding a document. It is not designed to be applied on top of an existing, traditional TEI encoding. Links are not maintained between the textual content of the document and separately encoded layout information. For this reason, some TEI projects adopt a cumbersome double-encoding approach, with one encoding for the representation of the source text(s) and a second encoding for the documentary edition [1], which it would be desirable to avoid.

3.2 XSL:FO

One of our design goals is to offer a method that provides a description of how the content of an encoding should be presented. This mechanism must be capable of describing different rendering outputs of the same musical content. XSL:FO (eXtensible Stylesheet Language: Formatting Objects) appears to be useful in this context as it allows a set of rules to be specified for the transformation of the content of an encoded document using a defined page layout. For this purpose, it uses templates which are instantiated as often as necessary during processing, until the entire content is rendered. Using XSL:FO <block> elements for systems, staves, and layers, the general layout of pages containing music notation can be described. XSL:FO can define the margins of the page, padding between and size of systems and staves, and so on.

Despite its initial promise, because XSL:FO is content-agnostic it cannot be used to adjust the layout in response to the content as required by music notation. For instance, in opera or other equally large scores, it is quite common that only the staves of the active voices or instruments be present. This leads to variation in the size and content of systems that is only achievable in XSL:FO by providing a large number of separate templates for each distinct case. Additionally, these templates need to be called explicitly by the user, so that a fully automatic rendering of the content is no longer possible. Furthermore, XSL:FO does not provide mechanisms for capturing the coordinate information necessary for diplomatic transcription of the sources. For these reasons, a template-driven language such as XSL:FO is not suitable for the description of content-dependent layout.

3.3 Scalable Vector Graphics

Instead of using templates for laying out pages, a description of the already laid-out pages could be another possibility. A legitimate approach for this would be to use Scalable Vector Graphics (SVG) markup to describe individual pages. There are already processors that generate SVG output from MEI markup. The problem with using SVG, however, is that it makes it nearly impossible to maintain a connection to the logical content. Because the SVG markup represents the graphical primitives of music notation (lines, note head shapes, etc.) and not the semantic information, changes in the content require the primitives to be recalculated. For example, the SVG markup for the representation of a beam would be made up of filled parallelogram shapes, one for each beam line, with their size and position on the page. Changing the pitch of a single note within the <beam> element in the MEI data would require the size and position of all the graphical components of the beam to be recomputed. Since SVG describes already-processed data, it is inappropriate for storing layout information in a flexible way despite its utility as an output format.

4. THE MEI LAYOUT MODULE

As mentioned above, XSL:FO offers general instructions on how to process data, whereas SVG is more appropriate for already-processed data. The ideal solution for MEI lies between the two: a description of what is in a source, or what should appear on every page in a rendered edition, without duplicating the content and without requiring additional processing of the data.

4.1 General organization

A solution to this problem is to store the layout information in a dedicated sub-tree separate from the musical content. The sub-tree is represented by a <layoutGrp> element within the <music> element. It may contain an arbitrary number of <layout> elements, each of them describing a different visualization of the same musical content.

For example, for the case illustrated in Figure 1 with two sources A and B, the musical content of both sources will be encoded following the traditional approach of MEI, in a single hierarchy with <app> elements for encoding their differences. At the same time, each source will be described further by its own layout sub-tree, if need be in parallel with its related facsimile.

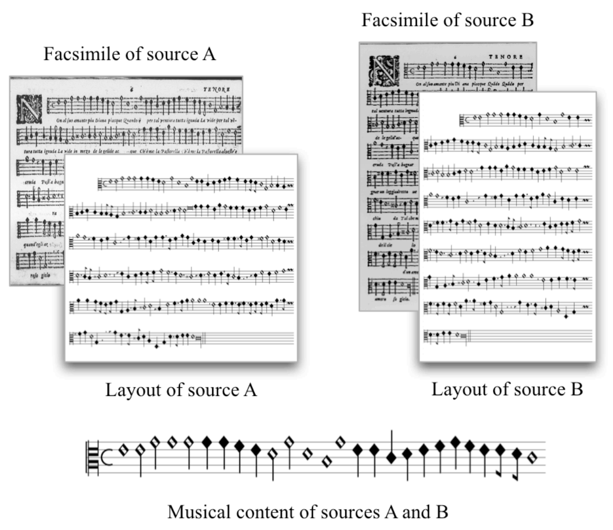


Figure 1. An example of two sources as organized with the layout module in MEI. While they share the same musical content, each layout is described in its own sub-tree.

The <layout> element is expected to have a @type attribute for indicating whether it is intended for "transcription" or "rendering". The <layout> element contains a sequence of <page> elements, each with page-level metadata and nesting <system>, <laidoutStaff> and <laidoutLayer> child elements that can precisely represent how each of them is positioned on the page. The hierarchy can be summarized as follows:

- layoutGrp
  - layout ('transcription' or 'rendering')
    - page
      - system
        - laidoutStaff
          - laidoutLayer

At the lowest level, the <laidoutLayer> element contains a list of <laidoutElement> children. Each <laidoutElement> acts as a generic container that can refer to any element within the corresponding <layer> element in the musical content sub-tree.

The <system>, <laidoutStaff>, <laidoutLayer> and <laidoutElement> elements all have attributes for storing their coordinate position (@lrx, @lry, @ulx and @uly) in "transcriptional" layouts.

4.2 Referencing system

The links that are established between the layout and the elements in the musical content sub-tree are a keystone of the module. Every <page> and <system> in the layout sub-tree is linked to its related <pg> and <sb> elements in the musical content sub-tree. In order to limit the modifications of the musical content sub-tree as much as possible, the links operate deliberately from the layout towards the content, and not the reverse. Each <page> element is expected to have a @pbrefs attribute with the list of XML IDs of <pb> elements in the musical content sub-tree to which it applies, as illustrated in Figure 2. Similarly, <system> elements have a @sbrefs attribute containing a list of <sb> elements. Therefore, the correct insertion of <pb> and <sb> elements is the only change to the logical tree required for this proposal.

For the <laidoutStaff> and <laidoutLayer> elements, the link with the musical content sub-tree is established using a @staff attribute that refers to the @n attribute of a <staff> element in the musical content sub-tree. Finally, <laidoutElement> elements have a @target attribute for referencing elements in the musical content sub-tree.

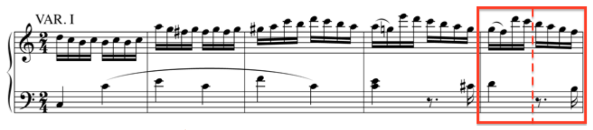
```
<music>
  <facsimile source="A">
    <!-- facsimile for source A -->
  </facsimile>
  <facsimile source="B">
    <!-- facsimile for source B -->
  </facsimile>
  <layoutGrp>
    <layout source="A" type="transcription">
      <page pbrefs="pb-A-1">
        <!-- the page layout in source A -->
      </page>
    </layout>
    <layout source="B" type="transcription">
      <page pbrefs="pb-B-1">
        <!-- the page layout in source B -->
      </page>
    </layout>
  </layoutGrp>
  <body>
    <mdiv>
      <score>
        <scoreDef barplace="mensur" key.sig="0">
          <staffGrp>
            <staffDef clef.shape="C" clef.line="3"/>
          </staffGrp>
        </scoreDef>
        <section>
          <staff n="1">
            <layer n="1">
              <pb xml:id="pb-A-1" source="A"/>
              <pb xml:id="pb-B-1" source="B"/>
              <sb xml:id="sb-A-1-1" source="A"/>
              <sb xml:id="sb-B-1-1" source="B"/>
              <!-- the musical content in A and B -->
            </layer>
          </staff>
        </section>
      </score>
    </mdiv>
  </body>
</music>
```

Figure 2. The scaffold encoding for the example given in Figure 1. The <pb> elements in the score are referenced from the <page> elements in the layout.

4.3 Overlapping hierarchies

As we have seen, a fundamental reason why it is advantageous to keep the layout information in a separate sub-tree is because the layout represents a distinct hierarchy that might overlap with the content hierarchy. A typical case is when a system break occurs in the middle of a measure. In such a situation, the same system is indicated in MEI by several <sb> elements, one in every layer where the system break occurs. The encoding in Figure 3 gives an example for such a case with a fictitious system break introduced in the middle of measure number five. In practice, this system break could be present in one or more sources, or it could be desired in a specific rendering. Notice that there are two <sb> elements, one for each layer.

As illustrated in Figure 4, in the corresponding layout sub-tree of this example, the second system references the two new <sb> elements via its @sbrefs attribute.



```
<measure n="5" xml:id="m5">
  <staff n="1" xml:id="m5s1">
    <layer n="1" xml:id="m5s1l1">
      <beam>
        <note xml:id="m5s1e1" pname="g" oct="5" dur="16"/>
        <note xml:id="m5s1e2" pname="f" oct="5" dur="16"/>
        <note xml:id="m5s1e3" pname="d" oct="6" dur="16"/>
        <note xml:id="m5s1e4" pname="c" oct="6" dur="16"/>
      </beam>
      <sb xml:id="sb-X-2-1" source="X"/>
      <beam>
        <note xml:id="m5s1e5" pname="b" oct="5" dur="16"/>
        <note xml:id="m5s1e6" pname="a" oct="5" dur="16"/>
        <note xml:id="m5s1e7" pname="g" oct="5" dur="16"/>
        <note xml:id="m5s1e8" pname="f" oct="5" dur="16"/>
      </beam>
    </layer>
  </staff>
  <staff n="2" xml:id="m5s2">
    <layer n="1" xml:id="m5s2l1">
      <note xml:id="m5s2e1" pname="d" oct="4" dur="4"/>
      <sb xml:id="sb-X-2-2" source="X"/>
      <rest xml:id="m5s2e2" dur="8" dots="1"/>
      <note xml:id="m5s2e3" pname="b" oct="3" dur="16"/>
    </layer>
  </staff>
  <slur staff="1" startid="#m5s1e1" endid="#m5s1e2"/>
</measure>
<sb xml:id="sb-Y-2-1" source=Y"/>
```

Figure 3. The customary encoding of measure 5 with an additional internal <sb>. The beginning of the new system is represented by two <sb> elements in the content sub-tree.

```
<page n="1">
  <system n="1">
    <laidOutStaff staff="1">
      <laidOutLayer>
        <!-- previous measures -->
        <!-- first half of measure 5 -->
        <!-- musical content up to the sb -->
      </laidOutLayer>
    </laidOutStaff>
    <laidOutStaff staff="2">
      <laidOutLayer>
        <!-- previous measures -->
        <!-- first half of measure 5 -->
        <!-- musical content up to the sb -->
      </laidOutLayer>
    </laidOutStaff>
  </system>
  <system n="2" sbrefs="sb-X-2-1 sb-X-2-2">
    <laidOutStaff staff="1">
      <laidOutLayer>
        <!-- second half of measure 5 -->
        <!-- musical content from the sb -->
        <!-- next measures -->
      </laidOutLayer>
    </laidOutStaff>
    <laidOutStaff staff="2">
      <laidOutLayer>
        <!-- second half of measure 5 -->
        <!-- musical content from the sb -->
        <!-- next measures -->
      </laidOutLayer>
    </laidOutStaff>
  </system>
</page>
```

Figure 4. The proposed encoded layout for the example given in Figure 3. The second <system> contains references to the two <sb> elements.

4.4 Content selection

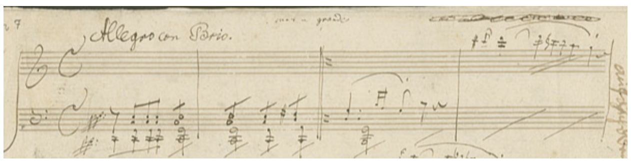
Implicitly, we expect the <laidoutLayer> element to include all elements contained in the corresponding <layer> element of the musical content sub-tree. This means that in the example illustrated by Figures 3 and 4, the content of the measure will be rendered implicitly up to the <sb> element for the system that ends in the middle of measure five and from the <sb> element for the next system. The use of <laidoutElement> for each element is optional for rendering layouts, but it is required for transcription layouts because in that case, we need to be able to store the coordinate positions of the elements.

In some cases, however, a more granular way of selecting content might be required. For example, it might be necessary to hide an element in a specific layout. For this purpose, the <laidoutElement> element has an @ignore attribute.

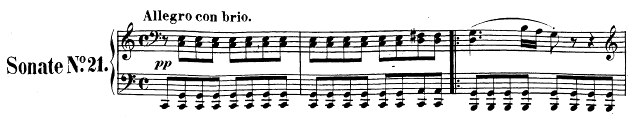
The selection of content can also be performed at the <laidoutStaff> level. For example, a layout for only one staff in the score will have only a single <laidoutStaff> element in each <system> element. Implicitly, all the other staves will not be included in that layout. Similarly, it is possible to change the order of the staves in a specific layout just by modifying the order of the <laidoutStaff> elements.

4.5 Textual and layout variants

In MEI, all variants are traditionally encoded with <app> and <rdg> elements in the music content sub-tree. In some cases, however, variants do not necessarily represent a textual difference between the sources because the musical content represented by the notation is identical. In Figure 5, we can see two examples of the beginning of Beethoven's "Waldstein" sonata. The right hand is written on the lower staff in the autograph and on the upper staff in the edition of Breitkopf & Härtel. Traditionally, this difference could be regarded as a variant in music critical editing even though the musical content is actually the same. However, it would not be possible to "hear" the difference between the two versions.



Autograph manuscript



Leipzig, Breitkopf & Härtel, (Serie 16, Plate B.144)

Figure 5. The beginning of Beethoven's "Waldstein" sonata No. 21. The right hand is written on the lower staff in the manuscript and on the upper one in the edition.

The layout module is designed in such a way that it is possible to encode purely presentational differences between sources at the layout level. In the <laidoutStaff> element, following a content selection method as described above, it is possible to retrieve content from another staff of the musical content sub-tree. In our example, this means that the lower <laidoutStaff> in the layout of the manuscript would pull the content from the first staff, assuming that the music content is encoded as in the edition. Even at its current experimental stage, this practice could represent a significant conceptual change in critical editing. The layout encoding itself becomes the way to represent layout variants, reserving the traditional <app> and <rdg> elements in the musical content sub-tree for textual differences.

5. CONCLUSION AND FUTURE WORK

We believe that the proposed solution is a novel method of encoding MEI documents because it creates a separation between the content of music notation and its possible realizations. Multiple realizations of the same musical content can be stored in parallel, each with its own specific layout information. The layout information can also provide additional functionality. It can be used for specifying how the content appears in an already-existing source, but it can also be used for specifying how the content must be rendered when creating a new edition. The first use is particularly interesting for OMR software applications and for producing image overlays for displaying a transcription directly on top of the facsimile image of the original source. The second use is particularly convenient for storing refined layout information for the parts of an encoded full score. The proposed module lays the basis for a new way of organizing the information contained in an existing MEI encoding.

This approach, however, also raises an interesting question regarding the line between content and presentation in music notation that we hope will receive more attention. There is clearly no fixed border because in music notation layout is a constituent component. The proposed layout module does not attempt to define an absolute boundary, but is intended to be flexible. In practice, the more varied the layout of the sources and the more detailed the layout information recorded, the less it will be desirable to keep layout information in the musical content sub-tree as has been MEI practice so far. With these changes, the musical content sub-tree may become a more abstract representation of the music.

The next step will be to finalize the module in preparation for the next official release of MEI, including preparing guidelines for its usage. We also expect to have to add more features at the <laidoutElement> level depending on thorough testing. For example, it would be logical to expect the module to handle the transposition of instruments when generating parts.

Because creating an encoding with multiple layouts in a general purpose XML editor will be unmanageable, tools that implement at least some of the features of this new module will be a high priority. Currently, the module is being implemented in the Aruspix software application, which will be used for prototyping and providing some more actual examples.

The authors believe that this proposal has great potential to enhance MEI's interoperability, to accelerate its further adoption by the scholarly community, and thus to reinforce its leading role in the digital humanities.

5.1 Availability

The module is available in the incubator of the MEI project.[1] It needs to be compiled with the Roma processor [5].

6. REFERENCES

[1] G. Brüning, K. Henzel, and D. Pravida: "Rationale of multiple encoding in the genetic Faust edition," *Journal of the Text Encoding Initiative*, [Forthcoming].

[2] M. Good and G. Actor: "Using MusicXML for file interchange," *Proceedings of the 3rd International Conference on WEB Delivering of Music,* p. 153, 2003.

[3] F. Jannidis (chair): "An encoding model for genetic editions," http://www.tei-c.org/Activities/Council/Working/tcw19.html 2011.

[4] A. Hankinson, L. Pugin, and I. Fujinaga: "An interchange format for optical music recognition applications," *Proceedings of the 11th International Conference on Music Information Retrieval,* pp. 51–56, 2010.

[5] A. Hankinson, P. Roland, and I. Fujinaga: "The music encoding initiative as a document-encoding framework," *Proceedings of the 12th International Conference on Music Information Retrieval,* pp. 293–298, 2011.

[6] L. A. Ludovico: "Key concepts of the IEEE 1599 standard," *Proceedings of the IEEE CS Conference: The Use of Symbols to Represent Music and Multimedia Objects,* p. 15–26, 2008.

[7] D. Oberhelman: "Quijote Interactivo (Interactive Quixote)," *Reference Reviews,* Vol. 25 No. 5, pp. 33–34, 2011.

[8] E. Selfridge-Field: *Beyond MIDI: The Handbook of Musical Codes*, MIT Press, Cambridge MA, 1997.

[1] <http://code.google.com/p/mei-incubator/source/browse/>