The Music Encoding Initiative (MEI)

Perry Roland
Digital Library Research & Development Group
Alderman Library, P.O. Box 400149
University of Virginia
Charlottesville, VA 22903-4149 USA
+1 434 982 2702
pdr4h@virginia.edu

ABSTRACT
This paper draws parallels between the Text Encoding Initiative (TEI) and the proposed Music Encoding Initiative (MEI), reviews existing design principles for music representations, and describes an eXtensible Markup Language (XML) document type definition (DTD) for modeling music notation which attempts to incorporate those principles.

Keywords
Text Encoding Initiative (TEI), Music Encoding Initiative (MEI), music notation, music representation, eXtensible Markup Language (XML), Document Type Definition (DTD) design

1   INTRODUCTION
The primary goal of the Text Encoding Initiative (TEI) was the creation of a comprehensive yet extensible standard for the encoding and transmission of textual documents in electronic form. However, the project's goals went beyond simple transcription. While not required to create a printed version of the text, elements that allowed authors to encode descriptive and interpretative information were seen as essential for computer-assisted analysis of an encoded text. The success of the TEI endeavor is demonstrated by the large and varied body of texts which have been encoded, many of which are available free of charge via the World Wide Web.

The need for a comprehensive standard for music has long been recognized. And now the possibility of a Music Encoding Initiative (MEI) is tantalizingly close. Since the beginning of TEI in 1987, SGML and its new incarnation XML have grown in popularity.   The development of HTML has made the creation and rapid growth of the World Wide Web possible. Domain-specific markup languages for both text and data, such as ThML for theological texts and Scalable Vector Graphics (SVG) for images, have been created in abundance. Yet no clear progress has been made in the encoding of music. Why is this?

2   SCOPE OF THE MEI
The first difficulty with advancing toward a TEI-like standard for music is identifying the proper scope of the encoding scheme.

In attempting to deal with this problem, the Standard Music Description Language (SMDL) chose to provide a markup language for any music from any locale and of any time. Despite achieving the status of an international standard, SMDL has not gained widespread acceptance, however, because it defines the term "music" much too broadly to effect a practical solution.

At first glance the TEI appears to have an overly broad scope as well. Text, just like music, has logical, visual, aural, and analytical aspects. However, the scope of the TEI is limited in a subtle yet significant way. The TEI was not designed to be the "human expression encoding initiative". Instead it is primarily concerned with the markup of those expressions which are, or can be, expressed as text; that is, those expressions which take a *written* form. Furthermore, the TEI is mute regarding the "proper" way to compose text. Even when texts are initially created using the TEI DTD, they are still essentially transcriptions of an ur-text.

Similarly, the MEI does not attempt to encode all musical expression, but instead limits itself to the *written* form of music, i.e. common music notation (CMN). Like the TEI, the MEI must also remain unconcerned with how music is created. It is not primarily an aid to musical composition just as the TEI does not function as an aid in the creation of text.

Some may see the adoption of CMN as the basis for encoding as too limiting. Legitimate arguments could be made for an entirely new form of music notation for the

purpose of electronic transcription. However, common music notation is applicable to a wide range of contemporary and, perhaps more importantly, historical music. It has been eloquently described by Selfridge-Field as "the cornerstone of all efforts to preserve a sense of the musical present for other and later performers and listeners" [9]. Given its expressiveness, extensibility, nearly universal usage, and longevity, there seems to be little reason not to adopt CMN as the starting point for the MEI.

The fact that the MEI fundamentally conceives of music as notation does not limit its usefulness for encoding performance and analytical information. While it cannot rival a human rendition, a basic performance suitable for many purposes may be mechanically derived from the notation. Of course, any additional information necessary to complete this process may also be encoded. Likewise, descriptive and critical information may be included to assist bibliographic and analytical applications.

Ultimately, a limited scope makes the design of a representation easier. For example, both the pitch and rhythm models can be greatly simplified when non-CMN requirements are not considered.

3 MEI DESIGN PRINCIPLES

The MEI relies upon basic design principles already identified as important for music representations.

Comprehensive
Simple codes which represent only the information necessary for a particular application seem to be more efficient because they require less development effort and less complex processing software; however, a comprehensive code is better able to capture the interdependency of the elements within a score. Also, because the simple encoding method requires a new encoding for each application, a comprehensive coding scheme is "more conducive to the establishment of a permanent data base of encoded musical scores, and to the ultimate prevention of duplicated effort" [11].

Declarative
Declarative representations, i.e. those which state knowledge *about* something, are preferred over procedural ones, i.e. those which state *how to do* something. Declarative knowledge can be examined and combined while procedural knowledge tends to be inaccessible. Declarative representations are more modular because they limit interactions between separate entities. In addition, knowledge can be added to a declarative representation easily while procedural representations allow addition only by modification followed by a required debugging process [2].

Explicit
In so far as possible, all relations and knowledge should be explicitly stated in the representation [2]. When a representation is declarative and explicit, it is naturally static. This is not to say that the representation cannot change. Indeed, the encoding may be the end result of a process, but the encoding does not represent the process, only the result.

Interpreted
In order to encode something, one must first determine what is to be represented by the encoding. The resulting encoding is an interpretation of the thing to be encoded [4]. In other words, whenever an attempt to assign meaning is made, interpretation occurs. A well-designed representation acknowledges this truism and allows one to make interpretations explicit.

Hierarchical
Many representation schemes have tended toward one of two extremes: encoding scores as a collection of notes or as a single entity without further division. However, there are many musically important structures that fall between these two extremes. The existence of multiple levels of structure implies the need for hierarchical representation.

There are several benefits in representing musical notation hierarchically. In a hierarchical representation, individual components are isolated, making it possible to limit interactions between components and to specify the scope of operators that act on them. In addition, this kind of structure allows any component of a score to be treated in exactly the same manner as any other, regardless of its size or position in the hierarchy. In other words, a hierarchical data structure is object-oriented. However, unnecessary complexity is not introduced by using a hierarchical structure instead of the list structure commonly used to represent music. A list structure can always be represented hierarchically if desired, i.e. by a tree having only 1 level [1].

Formal
A music representation should be as formal as possible. That is, the ability to prove the correctness of the encoding should not depend on knowledge outside the formal definition [2]. While it cannot model semantics, at least a DTD formally declares the syntax of the representation. A representation based on a formal grammar like that embodied in a DTD can describe a broad range of music. In fact, any music that can be segmented can be described by a grammar. Grammars are used extensively in a variety of disciplines. Therefore, a great deal of software, including parsers and compilers, has been developed around them [7].

Instead of attempting to accommodate archaic and incorrect practice, the MEI should continue efforts to modernize and standardize notational practices like those contained in Read [6].

Flexible
While a music representation should have a standardizing effect on the corpus for which it designed, it should not

strive to eliminate all variation. It must remain flexible enough to accommodate minor variations in the source material. In addition, much of the encoding should be made optional so that the encoder is not required to mark up things with which he is not concerned [4]. However, increased flexibility must be carefully weighed against an inevitable corresponding decrease in standardization.

Extensible

No representation can guarantee that it can be used for all future artifacts or anticipate all of its own future uses, especially if the representation is the result of a solitary effort. Therefore, extensibility is required. In addition, it may be necessary to extend the representation in order to resolve ambiguity already latent in the scheme [4].

Other principles

Additional desirable qualities -- unique, mnemonic, consistent, non-cryptic, non-context-dependent, and idiomatic -- have been described [4]. However, most of these qualities come for free with the adoption of XML as a basis for the representation. Therefore, they will not be discussed at length; however, these qualities will figure prominently in the evaluation of the DTD once it is completed.

4   DTD IMPLEMENTATION

It is not enough to simply identify desirable characteristics without putting them into practice. This section describes how the MEI DTD attempts to incorporate the design principles enumerated above.

Why not use an XML schema?

The choice to create an XML DTD instead of a schema is based on several factors. First, DTDs are a stable part of XML technology and are widely supported in XML software. Second, competing schema proposals make it difficult to predict when schemas will achieve the same widespread support. Third, and most importantly, because it does not include support for external entities, a schema is difficult to extend without completely rewriting it. The lack of external entities makes a schema, as well as markup instances, difficult to modularize. The sharing of markup fragments has valuable potential, in the creation of an album of musical compositions, for example. In short, the usefulness and longevity of schema are severely limited.

Elements versus Attributes

The decision regarding when to use elements and when to use attributes is more than a stylistic one. Each construct has advantages [3]. Elements are more useful when the encoded data requires structure, when it may have more than one value at a time, or when the data should make sense with the markup removed. Attributes, however, can be constrained by type and value and their values may be defaulted.

Rather than choosing to use one or the other exclusively, a moderate approach that takes advantage of the strengths of each is the best course. A mix of elements and attributes can help to illuminate the distinction between data that is structural, i.e. an object such as a note, and data that is a property or characteristic of the object, i.e. the pitch of the note. This kind of separation can prove valuable for later processing, but cannot be achieved using an all-element approach.

There are additional advantages to using attributes where possible. Because attribute values can be defaulted, using them instead of elements can eliminate unnecessary redundancy in the data and reduce the size of the encoded file, an important consideration in data transport. Attributes also make it easier to enforce one-to-one relationships, i.e. a single note can only have one pitch designation. In addition, since attributes may be constrained by value, their use encourages the creation of standardized markup.

Furthermore, attributes offer several processing advantages. Since attributes never contain structure, their use can make it possible to have the same element hierarchy to some arbitrary level in the hierarchy, leading to a tree that is more balanced and, therefore, to more efficient processing by down-stream applications such as parsers. Attributes are also easier to access by XML applications because they are immediately available within the context of the element. Since accessing attributes does not require iterative and recursive processing, unlike embedded elements, using them may result in faster processing.

"Milestone" elements

Since music notation contains many examples of multiple hierarchies, i.e. a beam that begins in one measure and ends in the next, it is tempting to use processing instructions or "milestone" elements with empty content models to represent them:

```
<measure>
      <note />
      <?beambegin ?>
      <note />
</measure>
<measure>
      <note />
      <?beamend ?>
      <note />
</measure>
```
<div align="center">Example 1</div>

There are significant disadvantages to this "pseudo-container" approach, however [10]. First, XML applications do not usually support processing instructions. Even if processing instructions were supported, however, it would be impossible to know exactly where the beam begins and ends. Does it begin with the first or second note in the first measure? We could *assume* that the beambegin

instruction precedes the first note lying under the beam, but this is only an assumption. In other words, this approach violates the principle that an encoding be explicit. Second, both constructs contradict the efficient elements-for-objects design described above. Last, the use of "milestone" elements is prone to error. It would be very difficult for a processing application to recover if the beamend construct were mistakenly omitted.

As seen in the example below, an element with a required participant list provides a better, that is, more XML-idiomatic, solution.

```
<measure>
      <note id="n1" />
      <beam corresp="n2 n3" />
      <note id="n2" />
</measure>
<measure>
      <note id="n3" />
      <note id="n4" />
</measure>
```
<div align="center">Example 2</div>

Flexibility
For the purpose of this discussion flexibility is defined as the ability to create restricted arrangements of the data to fit a specific purpose.

The adoption of an exchange DTD which is large and loosely structured does not preclude the creation of customized DTD subsets which are smaller, more tightly structured, and easier to learn and implement. In fact, for the DTD to be effective it must actually encourage this kind of use.

The MEI DTD attempts to achieve flexibility via several techniques. First, sub-element order is not prescribed in many element content models. Of course, attributes are never ordered. Second, many attributes have implied values, meaning they can be left out of an encoding when not needed. Next, many elements may be classed via their type attribute according to their role, effectively allowing them to have user-defined semantics. In addition, both the element's content model and attribute list may be restricted via the m.[element name] and a.[element name] parameter entities. Furthermore, an element may be "switched off" through the use of a marked section. Finally, even the name of the element may be changed via the n.[element name] parameter entity.

```
<!ENTITY % n.beam "beam">
<!ENTITY %n.beam; "INCLUDE">
<![%beam;[
<!ENTITY % m.x.beam "">
<!ENTITY % a.x.beam "">
<!ENTITY % m.beam
"((%n.note;|%n.chord;|%n.pad;|%n.rest;)+
```

```
%m.x.beam;))">
<!ENTITY % a.beam
"%a.common;
meiform        CDATA          #FIXED "beam"
breaksec       CDATA          #IMPLIED
rend           %beam.rend;    #IMPLIED
with           %beam.with;    #IMPLIED
%a.x.beam;">
<!ELEMENT %n.beam; %m.beam;>
<!ATTLIST %n.beam; %a.beam;>
]]>
```
<div align="center">Example 3</div>

Extensibility
As mentioned earlier, extensibility must be a primary concern for any representation that hopes to have a shelf life beyond the completion of the design phase. The MEI DTD employs parameter entities that allow users to add extensions. In addition to showing how an element can be restricted, the DTD fragment above illustrates how an element's content model and its attribute list may be extended. The m.x.[element name] and a.x.[element name] parameter entities may be used to add new sub-elements and attributes. Attribute value lists may also be extended with parameter entities.

Utilization of existing standards
Because progress toward an encoding standard for music notation is much more feasible when not locked into constant re-invention of past wheels, large parts of the design of the MEI DTD are drawn from existing standards.

On the largest scale, the MEI is modeled upon the TEI. At lower levels, the Acoustical Society of America (ASA) system is used to record pitch information, performance-specific data is encoded using elements which have similar names and functions as those in the Musical Instrument Digital Interface (MIDI) standard, most of the mark up for text is designed to be familiar to users of HTML, and TEI header and Dublin Core elements form the basis of the meta-data components. Of course, the Unicode standard underlies the character encoding model for XML, obviating the need to re-invent special character encoding schemes. Finally, while it is not a formal standard, a well-known, authoritative source [6] has been used as the basis for the grammar for music notation parts of the MEI.

5   FURTHER EVALUATION
Up to this point, the design of the MEI DTD has been a solitary effort. Of course, much more evaluation, perhaps against the DTD analysis principles identified by Megginson [5], will be required before it can become an industry standard.

The latest version of the DTD and accompanying markup examples are publicly available [8]. Comments are welcomed.

REFERENCES

1. Buxton, William, et al. The Use of Hierarchy and Instance in a Data Structure for Computer Music in <u>Foundations</u> <u>of</u> <u>Computer</u> <u>Music</u>. Curtis Roads and John Strawn, eds. Cambridge, MA: MIT Press, 1985.

2. Desain, Peter and Henkjan Honing. Issues in the Representation of Time and Structure in Music in <u>Music</u>, <u>Mind</u> <u>and</u> <u>Machine</u>: <u>Studies</u> <u>in</u> <u>Computer</u> <u>Music</u>, <u>Music</u> <u>Cognition</u> <u>and</u> <u>Artificial</u> <u>Intelligence</u>. Amsterdam: Thesis Publishers, 1992.

3. Graves, Mark. <u>Designing</u> <u>XML</u> <u>Databases</u>. Upper Saddle River, NJ: Prentice Hall PTR, 2002.

4. Huron, David. Design Principles in Computer-based Music Representation in <u>Computer</u> <u>Representations</u> <u>and</u> <u>Models</u> <u>of</u> <u>Music</u>. Alan Marsden and Anthony Pople, eds. New York: Academic Press, 1992.

5. Megginson, David. <u>Structuring</u> <u>XML</u> <u>Documents</u>. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

6. Read, Gardner. <u>Music</u> <u>Notation</u>: <u>A</u> <u>Manual</u> <u>of</u> <u>Modern</u> <u>Practice</u>. 2nd ed. New York: Taplinger, 1979.

7. Roads, Curtis. Grammars as Representations for Music in <u>Foundations</u> <u>of</u> <u>Computer</u> <u>Music</u>. Curtis Roads and John Stawn, eds. Cambridge, MA: MIT Press, 1985.

8. Roland, Perry. MEI WWW site: http://www.people.virginia.edu/~pdr4h/mei.

9. Selfridge-Field, Eleanor. <u>Beyond</u> <u>MIDI</u>: <u>The</u> <u>Handbook</u> <u>of</u> <u>Musical</u> <u>Codes</u>. Cambridge, MA: MIT Press, 1997.

10. St. Laurent, Simon. <u>XML</u> <u>Elements</u> <u>of</u> <u>Style</u>. New York: McGraw-Hill, 2000.

11. Wolff, Anthony B. Problems of Representation in Musical Computing. *Computers and the Humanities* 11 (1977), 3-11.