XML Project Notes

MusicXML

Definition:

MusicXML is an open source XML-based language used to reference western music. One of its main roles is sharing electronic sheet music. It can represent complex structured data, which is necessary when it comes to the intricacies of musical scores.

Industry Use:

MusicXML is, naturally, utilized within the music industry. Academics, such as musicologists, require access to many scores with as much ease as possible.

- Musicology apps and toolkits (MelodicMatch and music21)

Music software developers also make heavy use of MusicXML, which proves especially useful in composition and music production software. Here’s a list of companies and the software they’re affiliated with that incorporate MusicXML:

* MakeMusic (Finale, Finale Notepad, PrintMusic, Songwriter, and Dolet plug-ins for Finale and Sibelius)
* Sibelius (Sibelius Student and Sibelius First)
* Steinberg (Cubase and Nuendo)
* Apple - iOS Notation programs for iPhone and iPad (Symphony Pro, Notation, SeeScope, Maestro, Pocket Score)
* Note: While MusicXML is open-source, many of these programs are not. However, the creation of open source music software is encouraged on the official website.

MusicXML is also used in the selling of online sheet music.

* Sheet Music Sales Systems (Legato Sheet Music Viewer)
* Electronic Music Stands (Organ Muse)

History:

Before MusicXML, it was difficult to share music electronically. The only commonly supported music notation interchange format was MIDI. You may remember MIDI as the sound format that brought you those lovely synthesized versions of pop songs that used to auto-play if you visited a friend’s super cool page on Geocities. MIDI was serviceable for just that, performance applications, but it was weak when it came to actual music notation.

For example, MIDI was unable to tell the difference between a Gb and a F#. While those two notes are physically one and the same on a musical instrument (“enharmonic”), the context of the two notes are radically different and may even affect the tuning of a musical instrument when put into use. Aside from MIDI, there were two other predecessors to MusicXML: NIFF (NEED VISUAL EXAMPLE) and SMDL (NEED VISUAL EXAMPLE).

NIFF’s focus was graphical music representation. It had no inherent concept of, for example, the musical note “C”, but could place it on a picture of a staff. It worked well with scanning programs, but not at sequencing to a musical database, and at the end of the day, MIDI was still better than NIFF at both, though it still an edge on graphical notation.

SMDL was an ambitious venture that sought to represent every style of music that ever existed and ever could be created. It ended up being too complicated for the majority of people to understand, so was never given commercial software support.

Enter MusicXML. (NEED DATE OF CONCEPTION HERE.) MusicXML’s design comes from the Muse Data format (NEED VISUAL EXAMPLE), developed by Walter Hewlett (who is, in fact, tangentially related to the HP company) at the Center for Computer Assisted Research in the Humanities at Stanford University, and the Humdrum format (NEED VISUAL EXAMPLE) developed by David Huron at Ohio State University. The first beta version of MusicXML was essentially an XML (based on SGML and HTML) update of MuseData with some Humdrum components thrown in. There was a key difference in desired application, though. Both of the aforementioned formats dealt primarily with classical and folk music. MusicXML was intended for the pop contemporary genre.

(ANYTHING IN BETWEEN?)

For the past three years, a MusicXML community meeting has been taking place at the Musikmesse fair in Frankfurt, Germay.

As of July 28, 2015, the development of MusicXML has changed hands from MakeMusic to the W3C.

Example and Use:

(USE WIKIPEDIA FOR BASICS PAGE THEN DO SLIDE ON FAURE JUST TO SHOW HOW INTRICATE IT CAN GET).

MusicXML is used to express sheet music. In action, it’s language that values nesting elements over assigning attributes (even going as far as using an element called <attributes> to describe the set up for a score. Case in point: (Wikipedia example) is everything required to set up just a treble clef, a time signature, and one note. However, MusicXML uses language familiar to the music industry, so as wordy as the process is, the translation from music to text isn’t incomprehensible.

For example, after the prolog, which links to the MusicXML vocabulary, we come across the root element <score-partwise> followed by a version number, followed by the <part-list> element. This contains a list of the different instrumental or vocal parts in a piece. The parts are assigned an ID via the id attribute and named.

Now the actual music begins to happen. Inside a part, marked with an ID attribute that corresponds to the one assigned to it in the <score-part> element. From here, the information for the music is nested in <measure> elements, with a number attribute marking the measure’s order. Inside the measure is an <attributes> element, containing a <key> element. Nested in the <key> element is <fifths>, which holds a numeric value that relates to the position of a key in the Circle of Fifths . <time> is reserved for defining the time signature, and <clef> designates the desired clef and its position. Once the attributes are established, we get to the <note> element, which covers pitch (the name of the note and which octave its in) and duration (what kind of note and the length it’s held). To add to the basic elements for basic note display, there are also many elements related to the format of the page, the appearance of the staff, the appearance of the notes, how lyrics are placed, and metadata. When dealing with a full score, this amounts to thousands of lines of code for just one page of music.

<!-- It’s in the root element that we can also assign a title, and identification elements such as <creator> followed by the type attribute, so the difference between a composer and a lyricist can be noted. These only serve as metadata. The element <credits> covers the actual visual portrayal of the songwriters. There is also a <defaults> element that nests elements inside it pertaining to the visual layout of the score, dealing with height, width, margins, and more. <defaults> is not to be confused with <appearance> though, which deals with the appearance of the staff and notes themselves. The <credits> elements come at the end of the introductory section of code. 🡪

The Future:

As of September 7th, 2015, the short term goals for the development of MusicXML are to add support for SMuFL glyphs and identify the remaining gaps in adapting to compatibility with the rest of SMuFl. In the long term, there are plans to make a complete specification document and add DOM manipulation and interactivity to MusicXML.

April 8th will mark the first meeting of the W3C Music Notation Community Group in Europe (W3CMNCG?). One of the main focuses of the meeting will be on MusicXML 3.1 , which began development at the start of 2016.

Conclusion:

MusicXML has overtaken the music industry as the way to efficiently share music notation electronically. Though verbose, it is easy to understand and is widely supported by major music software programs, musicology apps, and electronic music sellers.