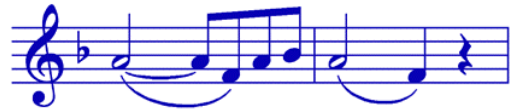
Recordare

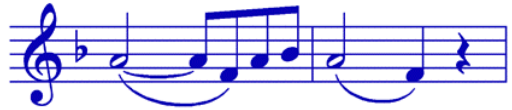
MusicXML: An Internet-Friendly Format for Sheet Music

Michael Good
Recordare LLC
good@recordare.com

Representing Music in Computer Formats



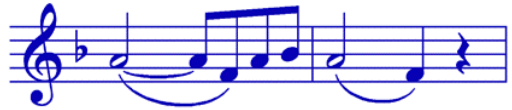
- Two main types of music data files
  - Audio: A recording of a musical performance, as in CDs and MP3
  - Symbolic: Underlying musical data (pitches, rhythms) as in MIDI
- This talk is about the symbolic data, as published in music notation
- Cannot create automatically from audio



Outline

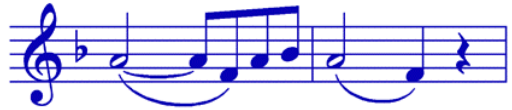
- The Need for a New Music Interchange Format
- MusicXML's Approach
- Elements of MusicXML Design
- MusicXML Development Environments
- Future Directions

The Need for a New Music Interchange Format

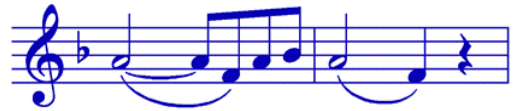


- Music notation publication has same great Internet potential as music audio, e-books, and other publications
  - Except that each music program has its own proprietary format
  - Or the music is published as PDF images with no musical semantics
- The only common interchange format, MIDI, does not meet publication needs

Prior Attempts at Moving Beyond MIDI



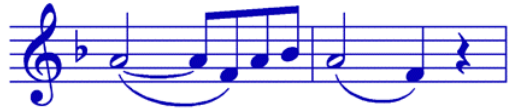
- NIFF
  - Represents music data graphically, with more notation data than MIDI
  - But worse than MIDI for performance and analysis applications
- SMDL
  - General-purpose music format
  - Overly complex; never implemented commercially



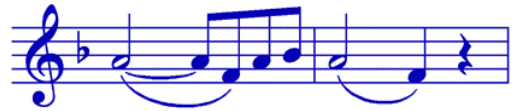
MusicXML's Approach

- A universal translator for common Western musical notation
- Supports notation, analysis, information retrieval, and performance applications
- Augments, but does not replace, specialized proprietary formats
- Adequate, not optimal, for diverse music applications

How to Succeed Where Many Have Failed

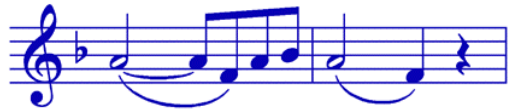


- We have an unfair advantage: XML
- MusicXML's design based on two powerful academic music formats: MuseData and Humdrum
- MusicXML definition developed iteratively with MusicXML software
- Support real music and real software

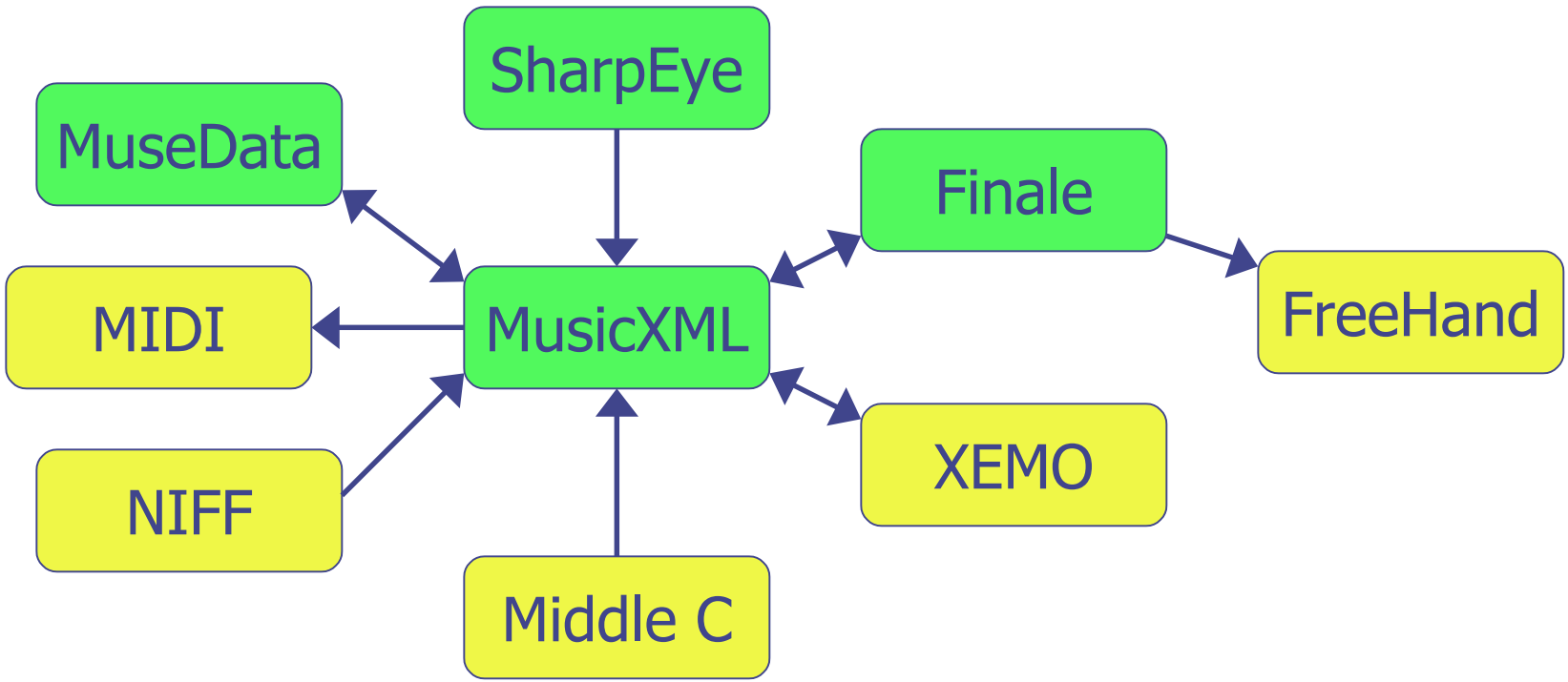


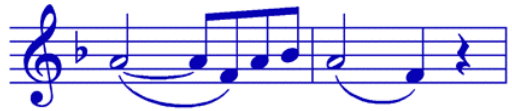
And It's Working

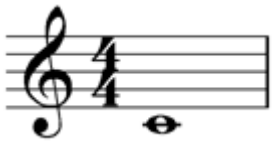
- MusicXML entered beta test in October 2001
- Beta software available now for Finale and SharpEye Music Reader
- Support announced from Xemus and Middle C Software
- Faster adoption than anything since MIDI



MusicXML Interchange

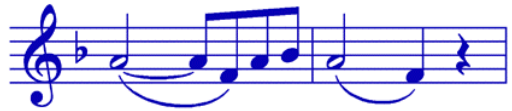


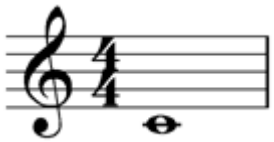




in MusicXML (1 of 2)

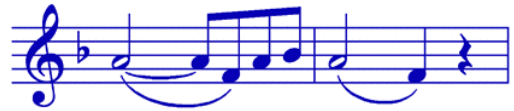
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
      "-//Recordare//DTD MusicXML 0.5a Partwise//EN"
      "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise>
   <part-list>
      <score-part id="P1">
        <part-name>Music</part-name>
      </score-part>
   </part-list>
   <part id="P1">
      <measure number="1">
```





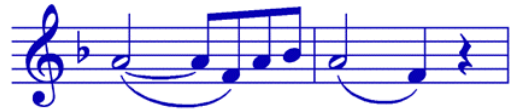
in MusicXML (2 of 2)

```
<attributes>
  <divisions>1</divisions>
  <key>
    <fifths>0</fifths>
  </key>
  <time>
    <beats>4</beats>
    <beat-type>4</beat-type>
  </time>
  <clef>
    <sign>G</sign>
    <line>2</line>
  </clef>
  </attributes>
  <note>
    <pitch>
      <step>C</step>
      <octave>4</octave>
    </pitch>
    <duration>4</duration>
    <type>whole</type>
  </note>
 </measure>
</part>
</score-partwise>
```



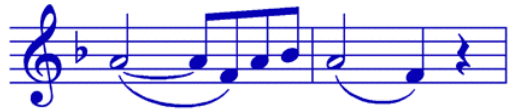
Elements of MusicXML Design

- Music is represented 2-dimensionally
  - Partwise DTD for measures within parts
  - Timewise DTD for parts within measures
  - XSLT stylesheets convert back and forth
- Separate elements for each part of musical semantics
- Applications can start by just supporting the basic MIDI-compatible elements

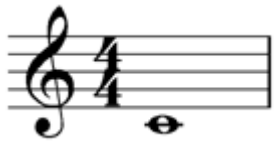


What MusicXML Leaves Out

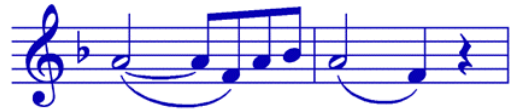
- MusicXML represents underlying musical data, not its interpretation
- It does not represent any individual music performance or engraving
  - No details about page and system breaks or absolute page positioning
  - No detailed MIDI performance data

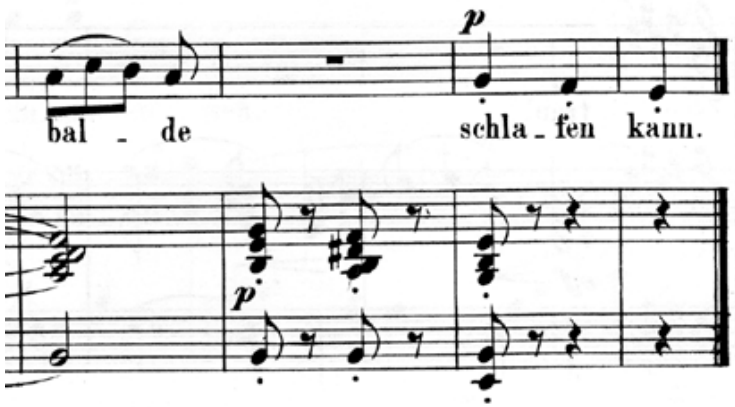


MusicXML vs. NIFF

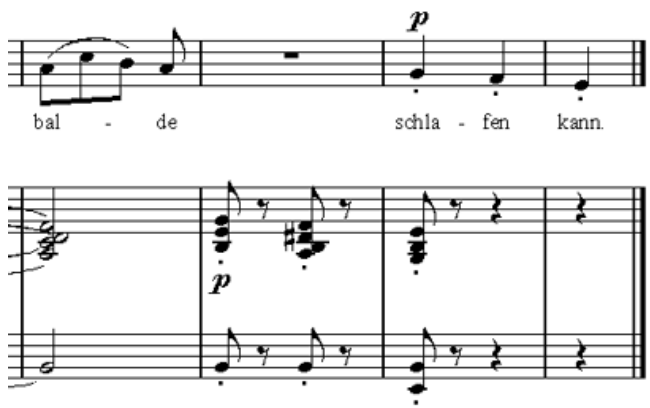
- How would  look in NIFF?
- Pitch is not represented in NIFF
- Instead, the notehead position is represented as a staff step of -2
- So to get the pitch we need to look up the clef, key signature, any prior accidentals, ties from accidentals, and 8^{va} marks

MusicXML vs. MIDI

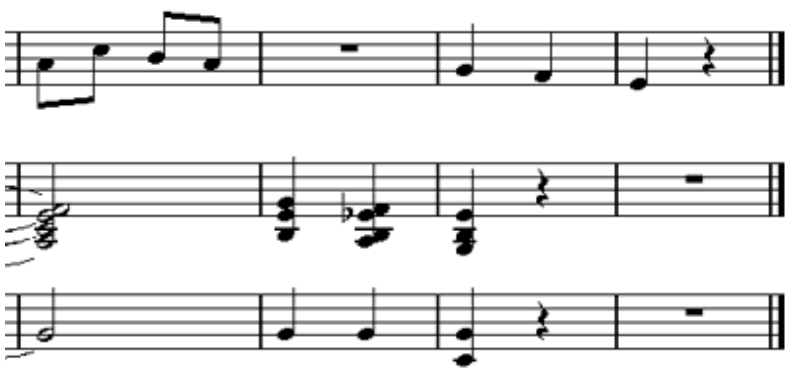




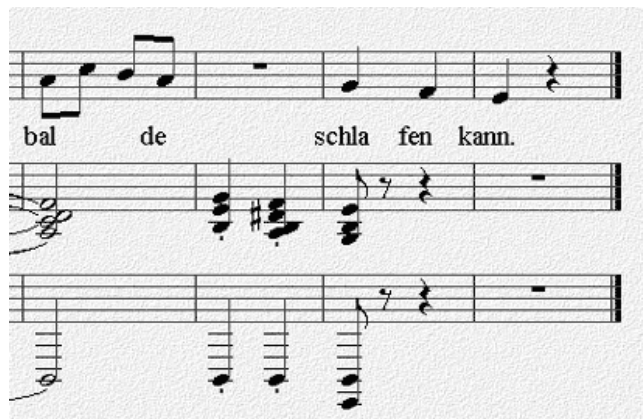
Original as scanned into SharpEye



Imported into Finale via MusicXML

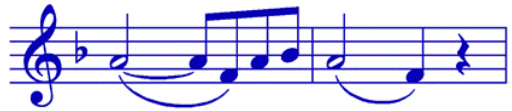


Imported into Finale via MIDI

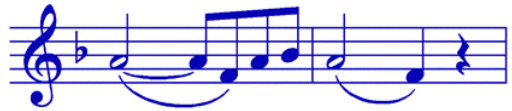


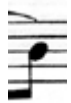
Imported into Sibelius via MIDI

Limitations of MIDI for Music Notation

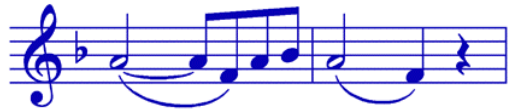


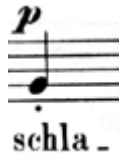
- MIDI represents NoteOn and NoteOff events with timestamps
  - Importer has to guess note durations
- MIDI represents keyboard keys
  - Importer has to guess E-flat or D-sharp
- MIDI does not include slurs, barlines, dynamics, clefs, beams, stems, …
- MusicXML represents all of this explicitly as XML elements – no guessing needed



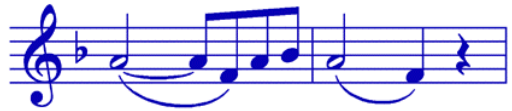
MusicXML for 

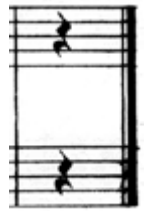
```
<!-- Divisions set to 6 earlier -->
<note>
   <pitch>
      <step>B</step>
      <octave>4</octave>
   </pitch>
   <duration>3</duration>
   <voice>1</voice>
   <type>eighth</type>
   <stem>down</stem>
   <beam number="1">end</beam>
   <notations>
      <slur type="stop" number="1" orientation="over"/>
   </notations>
</note>
```



MusicXML for 


```
<direction placement="above">
  <direction-type>
    <dynamics>
      <p/>
    </dynamics>
  </direction-type>
</direction>
<note>
  <pitch>
    <step>G</step>
    <octave>4</octave>
  </pitch>
  <duration>6</duration>
  <voice>1</voice>
  <type>quarter</type>
  <stem>up</stem>
  <notations>
    <articulations>
      <staccato/>
    </articulations>
  </notations>
  <lyric>
    <syllabic>begin</syllabic>
    <text>schla</text>
  </lyric>
</note>
```

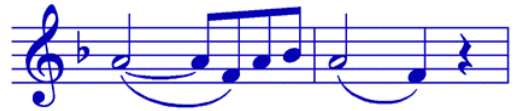


MusicXML for 

```
<measure number = "38">
    <note>
        <rest/>
        <duration>6</duration>
        <voice>1</voice>
        <type>quarter</type>
        <staff>1</staff>
    </note>
    <backup>
        <duration>6</duration>
    </backup>
```

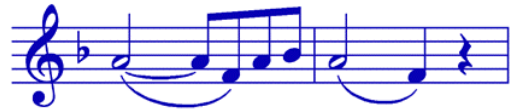
```
<note>
        <rest/>
        <duration>6</duration>
        <voice>2</voice>
        <type>quarter</type>
        <staff>2</staff>
    </note>
    <barline>
        <bar-style>light-heavy</bar-style>
    </barline>
</measure>
```

Freedom of Choice for Music Software Developers



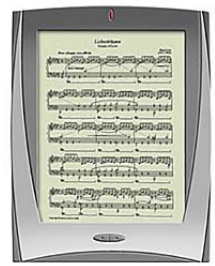
- Current music formats limit your choice of software tools:
  - Finale plug-ins require C or C++ code
  - Humdrum requires Unix knowledge
  - MuseData tools run on special TenX system
- Tight coupling of format and tools has hurt developer productivity

MusicXML Development Environments

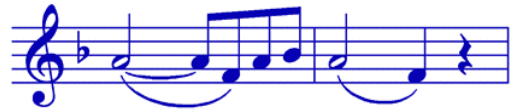


- XML promises that development can be done in many different environments
- MusicXML experience bears this out
  - Recordare software developed using Visual Basic and MSXML parser on Windows
  - Several developers using Java and Xerces parser on Linux, Mac OS X, and Windows
  - SharpEye software written in C, no parser

New Types of Music Use



- Electronic music stands and portable music displays
- Intelligent music accompaniment
- Music information retrieval
  - Query by humming
  - Include music semantics when searching for new music I might like
  - See http://music-ir.org



Future Directions

- Music information retrieval using XML databases and queries
- Address security issues using digital signatures or other DRM/PKI technology
- Continued adoption by music applications
- Standardization, probably via OASIS, after more implementation experience