An Extensive **MusicXML 2.0 Test Suite**

Reinhold Kainhofer, reinhold@kainhofer.com

Vienna University of Technology, http://www.fam.tuwien.ac.at/
GNU LilyPond, http://www.lilypond.org/
Edition Kainhofer, Music publishing, http://www.edition-kainhofer.com/

7th International Symposium on Computer Music Modeling and Retrieval,
Málaga, Spain
June 21-24, 2010

What is MusicXML?

- XML format to represent western-style music notation
  - Musical content (Notes, chors, dynamics, time, key, clef, etc.)
  - Exact page layout (MusicXML 2.0)
  - Audio representation (like MIDI, not performance recording)
- Defined originally via Document Type Definition (DTD) files and later also via XML Schema (XSD) files.
- Defined by Recordare LLC, plugins for Finale, Sibelius, etc.
- Supported (import and/or export) by many applications (notation, scanning, sequencers, etc.)

An example: Schubert's Ave Maria (excerpt)

```
<?xml version="1.0"
      encoding="UTF-8"?>
<!DOCTYPE score-partwise
      PUBLIC [...] >
<score-partwise
      version="2.0">
 <work>
  <work-number>D.
        839</work-number>
  <work-title>Ave
        Maria</work-title>
 </work>
 <identification>
  <creator
        type="composer">F.
        Schubert</creator>
  <encoding>
   <software>Finale 2005
         for
         Windows</software>
  </encoding>
 </identification>
 <defaults>
[...]
  <music-font
        font-family="Maestro"
        font-size="18"/>
 </defaults>
 <part-list>
  <score-part id="P1">
   <part-name>Voice</part-name>
[...]
  </score-part>
[...]
 </part-list>
 <!-- ================ -->
```

```
<part id="P1">
 <measure number="1">
  <attributes>
   <divisions>48</divisions>
   <key>
    <fifths>-2</fifths>
    <mode>major</mode>
   </key>
   <time symbol="common">
    <beats>4</beats>
    <beat-type>4</beat-type>
   </time>
   <clef>
    <sign>G</sign>
    <line>2</line>
   </clef>
   <staff-details
         print-object="no"/>
  </attributes>
  <note>
   <rest/>
   <duration>192</duration>
   <voice>1</voice>
  </note>
 </measure>
<!-- ================ -->
 <measure number="2">
  <note>
   <rest/>
   <duration>192</duration>
   <voice>1</voice>
  </note>
 </measure>
<!-- ================ -->
```

```
<measure number="3" width="654">
 <print new-system="yes"/>
 <barline location="left">
  <bar-style>heavy-light</bar-style>
  <repeat direction="forward"/>
 </barline>
 <note default-x="122">
  <pitch>
   <step>B</step>
   <alter>-1</alter>
   <octave>4</octave>
  </pitch>
  <duration>72</duration>
  <voice>1</voice>
  <type>quarter</type>
  <dot/>
  <stem
        default-y="-55.5">down</stem>
  <lyric default-y="-82" number="1">
   <syllabic>begin</syllabic>
   <text>A</text>
  </lyric>
  <lyric default-y="-104"
        number="2">
   <syllabic>begin</syllabic>
   <text>A</text>
  </lyric>
  <lyric default-y="-127"
        number="3">
   <syllabic>begin</syllabic>
   <text>A</text>
  </lyric>
 </note>
 <note default-x="326">
```

Why a Test Suite

- No free reference implementation available (official advice: Use the proprietary Dolet plugin for Finale)
- No set of basic unit test files available
- Only comments in the specification, no other format documentation
- Only some complex sample files available at MusicXML homepage, showing off what MusicXML is able to do

Aim of this Unit Test Suite

- Full coverage including all possible elements and all combination not possible
- $\Rightarrow$ Create representative test cases to catch as many common combinations as possible
- Small test cases, where a bug in one feature does not influence other cases
- Cover also some less used musical notation elements (but no cross-influences with other elements)

Structure of the Test Suite

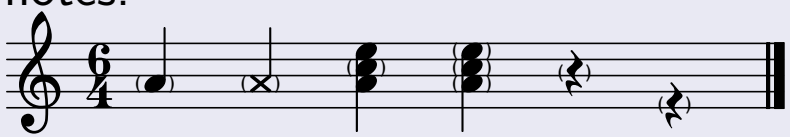
- Structured by file name!
- 12 large feature categories (separate aspects of MusicXML, e.g. basic musical notation, staff attributes, note-related elements, page layout, etc.)
    - Each category split into more specific aspects
        - Each such aspect gets several different, non-overlapping test cases
- More than 120 small unit test cases

Testing multiple possible element uses vs. separation of separate item

Example: Parenthesized noteheads (`<notehead parentheses=.../>`)

- Parenthesized normal noteheads
- Parenthesized non-standard noteheads
- Parenthesized noteheads inside a chord
- Parenthesized chords (all noteheads)
- Parenthesized rests (default position)
- Parenthesized rests (explicit position)

The test case `22d-Parenthesized-Noteheads.xml` for parenthesized noteheads tests all these cases in one file, but each of the settings on separate notes:

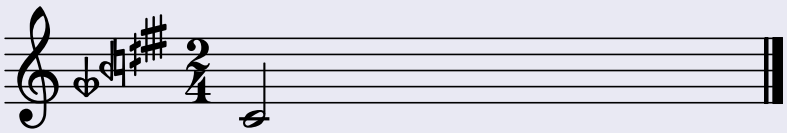


Exotic example: Key signatures with microtones (33b-Spanners-Tie.xml)

```
[...]

    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <key-step>4</key-step>
          <key-alter>-1.5</key-alter>
          <key-step>6</key-step>
          <key-alter>-0.5</key-alter>
          <key-step>0</key-step>
          <key-alter>0</key-alter>
          <key-step>1</key-step>
          <key-alter>0.5</key-alter>
          <key-step>3</key-step>
          <key-alter>1.5</key-alter>
        </key>
        <time>
          <beats>2</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>

[...]
```



- Very exotic case!
- All possible alterations are checked!
- Observe bad XML design (see later!)

Connection to LilyPond

- Originally: Some test files for `musicxml2ly` (Converter from MusicXML to LilyPond; `http://www.lilypond.org/`)
- Automated sample renderings can be done of MusicXML test case (No reference renderings!):
  - `musicxml2ly` is just one particular implementation with one particular interpretation of ambiguities!
  - `musicxml2ly` does not support every aspect perfectly
  - The MusicXML specification leaves many things open ($\Rightarrow$ left to each importing application!)
- Future plan: Include sample renderings from other applications, too. (Need to extend lilypond-book for this!)

Conclusion

- Finally a MusicXML test suite is freely available ($>$ 120 test files, MIT license):

Homepage of the test suite

`http://kainhofer.com/musicxml/`

- Sample renderings available (created via `musicxml2ly` and LilyPond)
- MusicXML is a good industry standard for music notation exchange
- Several minor issues in MusicXML format

Acknowledgements

- LilyPond developers and community!
- MusicXML mailing list (in particular Michael Good, author of the MusicXML specification)
- Vienna University of Technology, Christian Doppler Laboratory for Portfolio Risk Management (PRisMa Lab)