An extensive MusicXML 2.0 test suite

Reinhold Kainhofer

Vienna University of Technology,
Wiedner Hauptstraße 8-10/105-1, A-1040 Wien, Austria,
and
GNU LilyPond, http://www.lilypond.org/
and
Edition Kainhofer, Music Publishing, Wien, Austria
reinhold@kainhofer.com, http://reinhold.kainhofer.com/

Abstract. MusicXML [4] – a widely used interchange format for music data based on XML (Extensible Markup Language) – is specified using DTD (Document Type Definition) files and equivalently some XSD (XML Schema Document) files. However, up to date, no representative archive of MusicXML unit test files has been available for testing purposes. Here, we present such an extensive unit test suite. Although originally intended as regression testing tools for the MusicXML to LilyPond converter (musicxml2ly), it has turned into a general MusicXML test suite consisting of more than 120 MusicXML test files, each checking one particular aspect of the MusicXML specification. The connection to the LilyPond project also makes it possible to provide sample renderings of the test cases produced by LilyPond's MusicXML converter.
The test suite is available at: http://kainhofer.com/musicxml/

Key words: MusicXML, Test suite, Regression testing, Music data format

1 Introduction

MusicXML has become the de-facto exchange format for visual music data, supported by dozens of software applications as an import or export format, or even as their native data format. It has also been proposed for scientific uses like musicological analysis and for online-music editing. The newly published Open Score Format specification [5] also uses MusicXML as its data format.

Despite a full syntactic definition of the MusicXML format [4] by Recordare, no official suite of representative MusicXML test files has been available for developers implementing MusicXML support in their applications. The only help were the comments and explanations given in the specification to create test cases manually. The predominant advice is to use the Dolet plugin for Finale[1], which

[1] The MusicXML import and export functionality of the commercial music notation program "Finale" is provided through a plugin called "Dolet" and developed by Recordare. Finale itself includes only a limited version of this plugin, while complete MusicXML 2.0 support is available by the full Dolet 5 for Finale plugin available from Recordare separately.

is a proprietary Windows and MacOS application that is not easily available for Open Source developers working on Linux. Also, this approach invariably will lead MusicXML to be interpreted as behaving like the Dolet plugin instead of being an application-independent specification.

This lack of a complete set of MusicXML test files was our main incentive for creating the present test suite [2]. Due to the complexity of musical notation, a complete set of test cases covering every possible combination of notation and all possible combinations of XML attributes and elements is probably out of reach. However, we have attempted to create representative samples to catch as many common combinations as possible[2]. Our main goals were to create small unit test cases, covering not only the most common features of MusicXML, but also some less used musical notation elements, like complex time signatures, instrument specific markup, microtones, etc.

It is available for download at: `http://kainhofer.com/musicxml`

2 Structure of the test suite

We identified twelve different feature categories, each dealing with separate aspects of the MusicXML specification, and have further split them into more specific areas (see Table 2). An example of such a feature category would be "Staff attributes", while the more specific areas in that category are "Time signatues", "Keys signature" and "Clefs". For each of these we have created several test cases.

The categorization leaves room for further enhancements of MusicXML by not assigning all possible numbers used to identify feature categories. The choice of the feature categories is not related to the structure of the MusicXML specification into several (dependent) DTD files, but rather designed from the viewpoint of an implementor of MusicXML functionality.

The test suite currently consists of more than 120 test cases, each representing one particular aspect of the MusicXML format. The file name (for example "`13d-KeySignatures-Microtones.xml`") starts with two digits, encoding the area of the test (see Table 1 for the exact meaning of the first two digits), followed by a letter to enumerate the test cases within each category. Finally, a short verbal description[3] is given in the file name.

If a feature has multiple possible attribute values or different uses within a score, the corresponding test file contains several subtests, separated as much as possible by using different notes or even different measures or staves for each of the values or combinations.

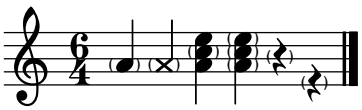
[2] For example, pitch information in MusicXML includes pitch, alteration and absolute octave information. However, some notation programs use relative mode, where the octave of a note is taken relative to the previous note. For these applications, it is not sufficient to simply check some notes with given absolute pitch, but also provide a test case for different intervals between two subsequent notes.

[3] A more detailed description is given inside the XML file in a `<miscellaneous-field name="description">` element.

01-09 ... Basics
01 Pitches
02 Rests
03 Rhythm
10-19 ... Staff attributes
11 Time signatures
12 Clefs
13 Key signatures
20-29 ... Note-related elements
21 Chorded notes
22 Note settings, heads, etc.
23 Triplets, Tuplets
24 Grace notes
30-39 ... Dynamics, artic., spanners
31 Dynamics and other single symbols
32 Notations and Articulations
33 Spanners
40-44 ... Parts
41 Multiple parts (staves)
42 Multiple voices per staff
43 One part on multiple staves

45-49 ... Measures and repeats
45 Repeats
46 Barlines, Measures
50-54 ... Page-related issues
51 Header information
52 Page layout
55-59 ... Exact positioning
60-69 ... Vocal music
61 Lyrics
70-75 ... Instrument-specific
71 Guitar notation
72 Transposing instruments
73 Percussion
74 Figured bass
75 Other instrumental notation
80-89 ... MIDI and sound
90-99 ... Other aspects
90 Compressed MusicXML files
99 Compat. with broken MusicXML

Table 1. Structure of the files, categorized by file name. Not all numbers are assigned to leave room for future extensions of the test suite and the MusicXML specification.



For example, parenthesized notes or rests use the `parentheses` attribute of a `notehead` XML element. However, for an application it might make a difference if that note is a note on its own, a note with a non-standard note head or if it is part of a chord. Similarly, rests can have a default position in the staff or an explicit staff position. The test case for parenthesizing (shown on the right) covers all these cases.

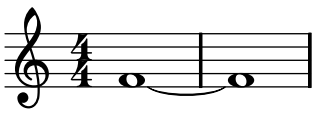
For practical reasons, the number of files in a test suite should not be too large to allow also manual checking of test cases. On the other hand, if too many different cases or attribute combinations are included in the same file, bugs in one feature area can influence test results for other features. This makes is harder to find the actual cause of a problem and deviates from the idea of unit tests.

One of the main aims for the test suite was thus to provide a good balance between a relatively low number of test files, while still clearly separating test cases for different features to avoid the described influences.

3 Examples of unit tests

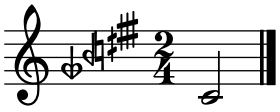
The unit test files are kept as simple as possible, so that they can best fulfil their purpose of checking only one particular aspect. For example, the unit test file `33b-Spanners-Tie.xml` for simple ties – a trivial feature, which still needs to be tested in regression tests – reads:

```
<?xml version="1.0"
    encoding="ISO-8859-1"
    standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML
    0.6b Partwise//EN" [...] >
<score-partwise>
 <identification>
  <miscellaneous>
   <miscellaneous-field
        name="description">Two tied
        notes</miscellaneous-field>
  </miscellaneous>
 </identification>
 <part-list><score-part
      id="P1"/></part-list>
 <part id="P1">
  <measure number="1">
   <attributes>
    <divisions>1</divisions>
    <key><fifths>0</fifths></key>
    <time>
     <beats>4</beats>
     <beat-type>4</beat-type>
    </time>
    <staves>1</staves>
    <clef number="1">
     <sign>G</sign>
     <line>2</line>
    </clef>
   </attributes>
   <note>
    <pitch>
     <step>F</step>
     <octave>4</octave>
    </pitch>
    <duration>4</duration>
    <tie type="start"/>
    <voice>1</voice>
    <type>whole</type>
    <notations><tied
        type="start"/></notations>
   </note>
  </measure>
  <measure number="2">
   <note>
    <pitch>
     <step>F</step>
     <octave>4</octave>
    </pitch>
    <duration>4</duration>
    <tie type="stop"/>
    <voice>1</voice>
    <type>whole</type>
    <notations><tied
        type="stop"/></notations>
   </note>
  </measure>
 </part>
</score-partwise>
```



Other test files check for more exotic MusicXML features, like for example the test case for non-traditional key signatures with microtone alterations (excerpt of `13d-KeySignatures-Microtones.xml`):

```
<attributes>
  <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>
</attributes>
```



However, in all cases the structure of the test file is kept as simple as possible to avoid cross-interactions of bugs in different areas of an application.

4 Sample renderings

Originally, the files of the test suite were generated as test cases for the implementation of musicxml2ly, a utility to convert MusicXML files into the LilyPond [3] format. Using this utility, sample renderings of all the test cases can be created automatically and are made available on the homepage of the test suite.

However, as the musicxml2ly converter does not yet support every aspect of MusicXML correctly, these renderings cannot be regarded as official reference renderings, but rather as an indication how one particular application understands the files.

5 License of the test suite

As one of the purposes of a test suite for a standard is that it can be freely used for testing purposes of any application or tool using this standard, the MIT license was chosen for the test suite. This means that there are no usage restrictions and the set of test files can be freely used for any purpose, as long as the LICENSE file (or the copyright notice) is preserved.

6 Conclusion

The MusicXML test suite presented here provides software developers in the area of music notation with an extensive set of representative test cases to check conformance to the MusicXML specification and to perform regression and coverage tests.

Even though MusicXML has established itself as an industry standard for the extremely hard and complex task of exchanging music notation, it is still encumbered with some minor issues. The problems we encountered while creating the test suite were reported and discussed with the authors of the MusicXML specification on the semi-public MusicXML mailing list.

One can expect that future versions of MusicXML will solve or at least soften most of the issues we encountered.

References

1. M. Good. Lessons from the adoption of MusicXML as an interchange standard. In *Proc. XML 2006*, 2006.
2. R. Kainhofer. Unofficial MusicXML test suite. `http://kainhofer.com/musicxml/`, 2009. Representative set of MusicXML test cases.
3. H.-W. Nienhuys and J. N. et al. GNU LilyPond. `http://www.lilypond.org/`, 2010.
4. Recordare LLC. MusicXML 2.0. Document Type Definition (DTD): `http://musicxml.org/dtds`, XML Schema Definition (XSD): `http://musicxml.org/xsd`, 2010.
5. Yamaha Corporation. Open Score Format (OSF, ver. 1.0), packaging specification, 2009. `http://openscoreformat.sf.net/`.