

MusicXML

craig@ccrma.stanford.edu

26 February 2013

Parameters

Fixed Function(int one, int two, int three)

C

Variable Function(const char* format, ...)

<http://cc.byexamples.com/2007/01/18/va-list-create-function-like-printf-2>

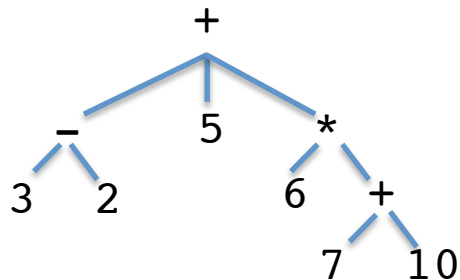
Optional Function(int one, int two = 2, int three=3)
(like Guido)

C++

Overloading Function(int one, int two)
Function(int three)

Tree (+ (- 3 2) 5 (* 6 (+ 7 10)))

Lisp



MIDI Parameters

- All MIDI protocol parameters are **fixed** excepts for “system exclusive messages”
- Meta messages (component of MIDI files, not MIDI protocol) are **variable**.

0x90 60 127

note(channel, key, velocity)

0xE6 0x7f 0x7f

bend(channel, LSB, MSB)

- Allows hot-plugging of MIDI cable.
- Limits expandability (function space maximized with fixed parameter commands)

SCORE Parameters

- SCORE items are all **variable** length fixed parameter lists.
- Similar to MIDI meta message system
- Identical to Music V (C Sound) parameter system

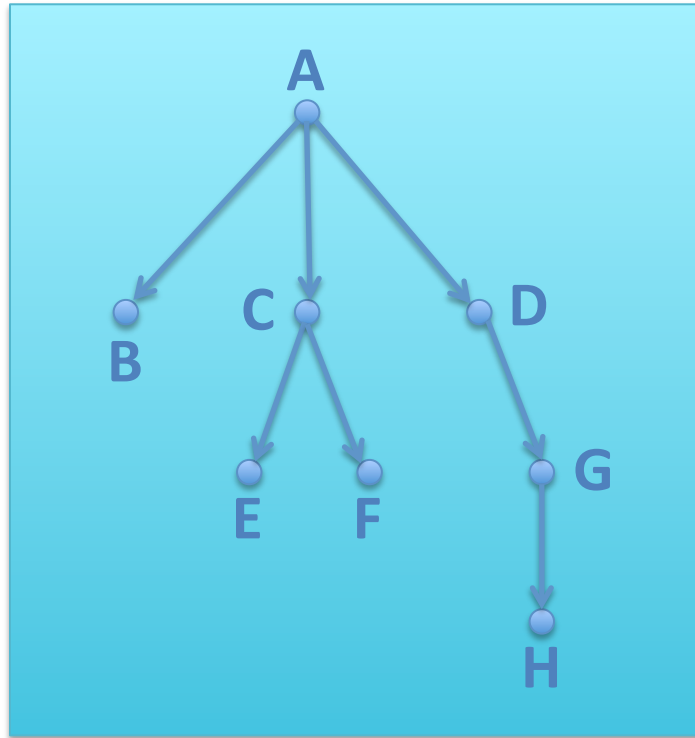
<http://www.csounds.com/chapter1/index.html>

```
8 1 0 0 0.6 128.146
14 1 0 3
3 1 1.2 0 0.8
17 1 5.997 0 -1
1 1 9.297 7 20 1 2
1 1 20.566 4 10 2 4
1 1 50.64 8 20 1 2
5 1 50.64 8.5 8.5 64.016 1.579 -2
14 1 61.923 1
1 1 64.016 8 20 1 2
1 1 75.291 6 10 1 2
1 1 86.561 5 10 2 4
14 1 109.113 1
1 1 111.206 9 20 2 4
14 1 128.146 1 3
```

- Allows for both forwards and backwards compatibility:
 - New parameters added to end of current list
 - Old program ignores (but preserves) unknown parameters.

XML

- XML describes a tree structure:



- Serialization showing hierarchy:

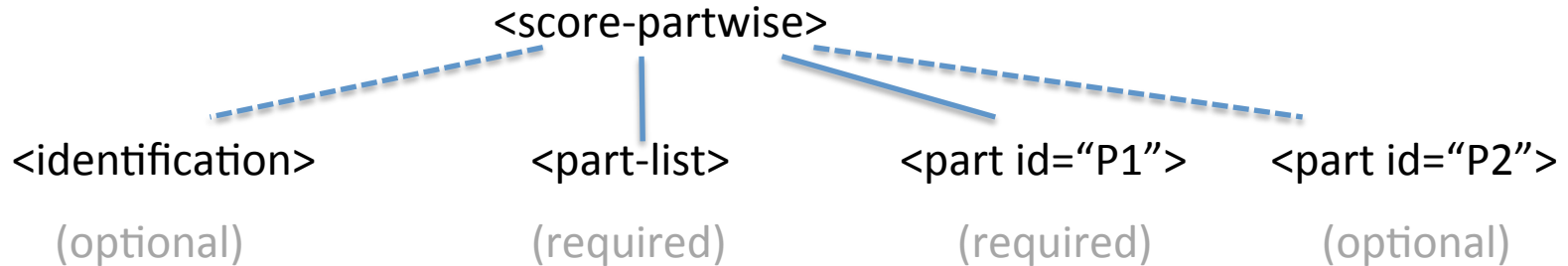
```
<A>  
  <B/>  
  <C>  
    <E/>  
    <F/>  
  </C>  
  <D>  
    <G>  
      <H/>  
    </G>  
  </D>  
</A>
```

- Equivalent serialization: `<A><C><E/><F/></C><D><G><H/></G></D>`
- LISP analogy: `(a b (c e f) (d (g h)))`

XML

- Advantage: Simple parsing model for data storage
 - Like MIDI, SCORE, LISP
 - Unlike Guido, Lilypond, C, C++, Java, JavaScript (lex/bison type formats)
- Allows for hierarchical structuring of data
 - Good: music notation usually fits well into hierarchical model
 - Useful for manipulating music
 - Bad: music notation is 2-dimensional, XML is 1-dimensional (superposition of multiple hierarchies)
- Allows for forwards compatibility, and backwards compatibility if careful
 - Possible to add new parameters without altering parsing

MusicXML Data hierarchy (root)



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 1.0 Partwise//EN"
    "http://www.musicxml.org/dtds/1.0/partwise.dtd">
<score-partwise>
```

<score-partwise> is the *root element*

```
<!ELEMENT score-partwise (%score-header;, part+)>
```

```
<!ENTITY % score-header
    "(work?, movement-number?, movement-title?,
    identification?, defaults?, credit*, part-list)">
```

DTD/Schema

DTD 

```
<!ELEMENT score-partwise (%score-header;, part+)>
```

Schema 

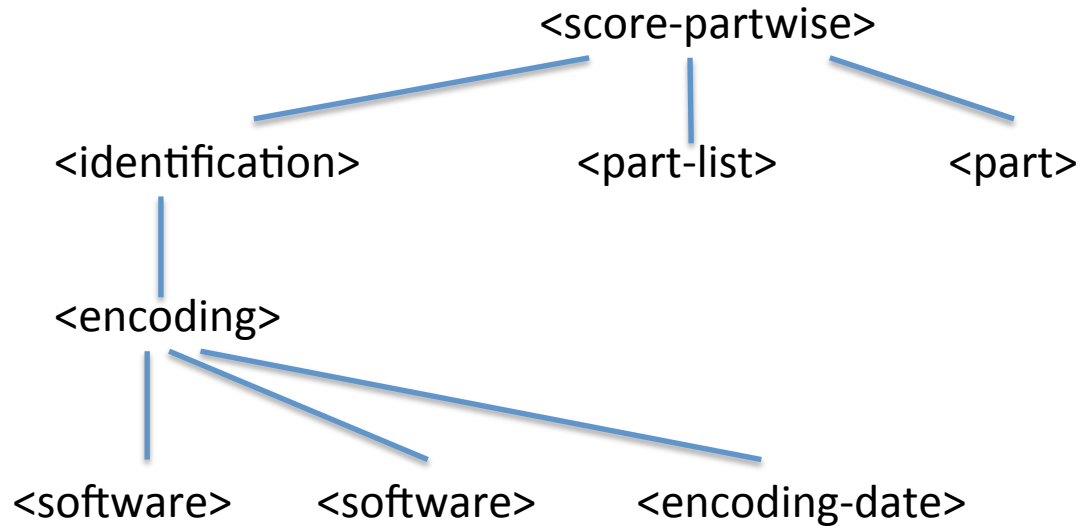
```
<!ENTITY % score-header
```

```
"(work?, movement-number?, movement-title?,  
identification?, defaults?, credit*, part-list)">
```

```
<xs:element name="score-partwise" block="extension substitution" final="#all">  
<xs:annotation>  
<xs:documentation>  
The score-partwise element is the root element for a partwise  
MusicXML score. It includes a score-header group followed by a  
series of parts with measures inside. The document-attributes  
attribute group includes the version attribute.  
</xs:documentation>  
</xs:annotation>  
<xs:complexType>  
<xs:sequence>  
<xs:group ref="score-header"/>  
<xs:element name="part" maxOccurs="unbounded">  
<xs:complexType>  
<xs:sequence>  
<xs:element name="measure" maxOccurs="unbounded">  
<xs:complexType>  
<xs:group ref="music-data"/>  
<xs:attributeGroup ref="measure-attributes"/>  
</xs:complexType>  
</xs:element>  
</xs:sequence>  
<xs:attributeGroup ref="part-attributes"/>  
</xs:complexType>  
</xs:element>  
</xs:sequence>  
<xs:attributeGroup ref="document-attributes"/>  
</xs:complexType>  
</xs:element>
```

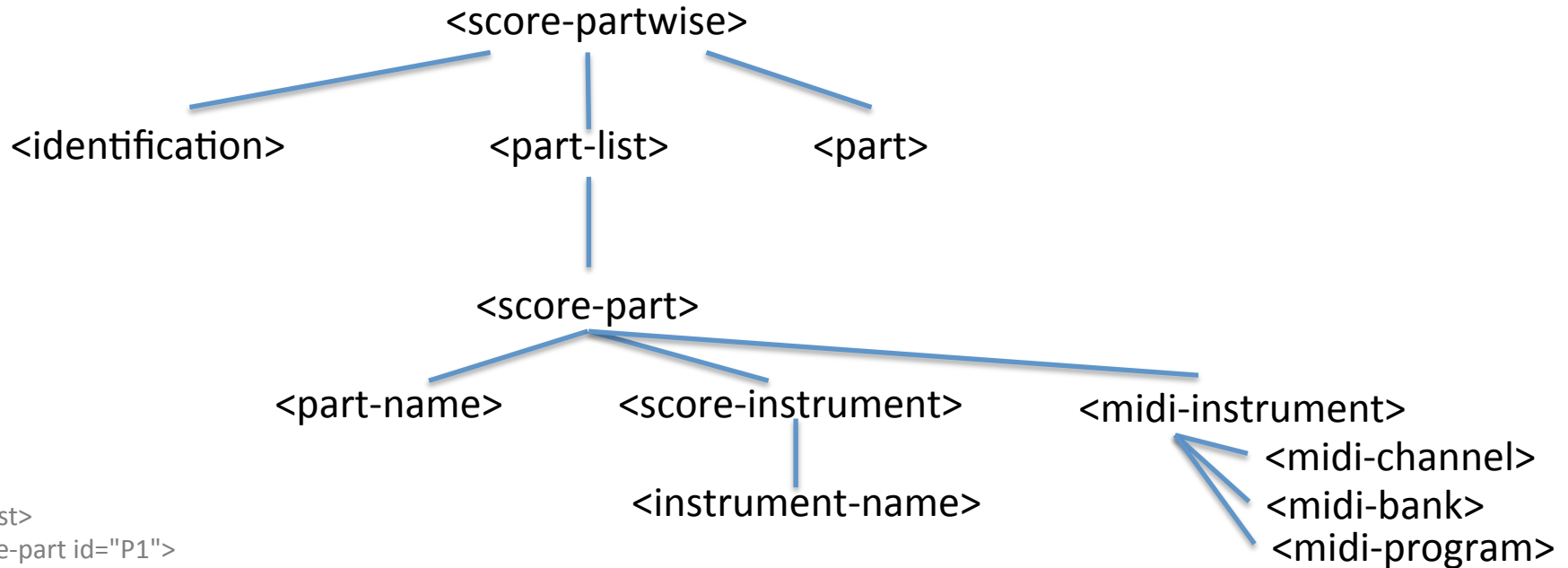
See: <http://www.makemusic.com/musicxml/specification/dtd>
(in ZIP: DTD and schema equivalent for MusicXML 3.0)

MusicXML Data hierarchy (header)



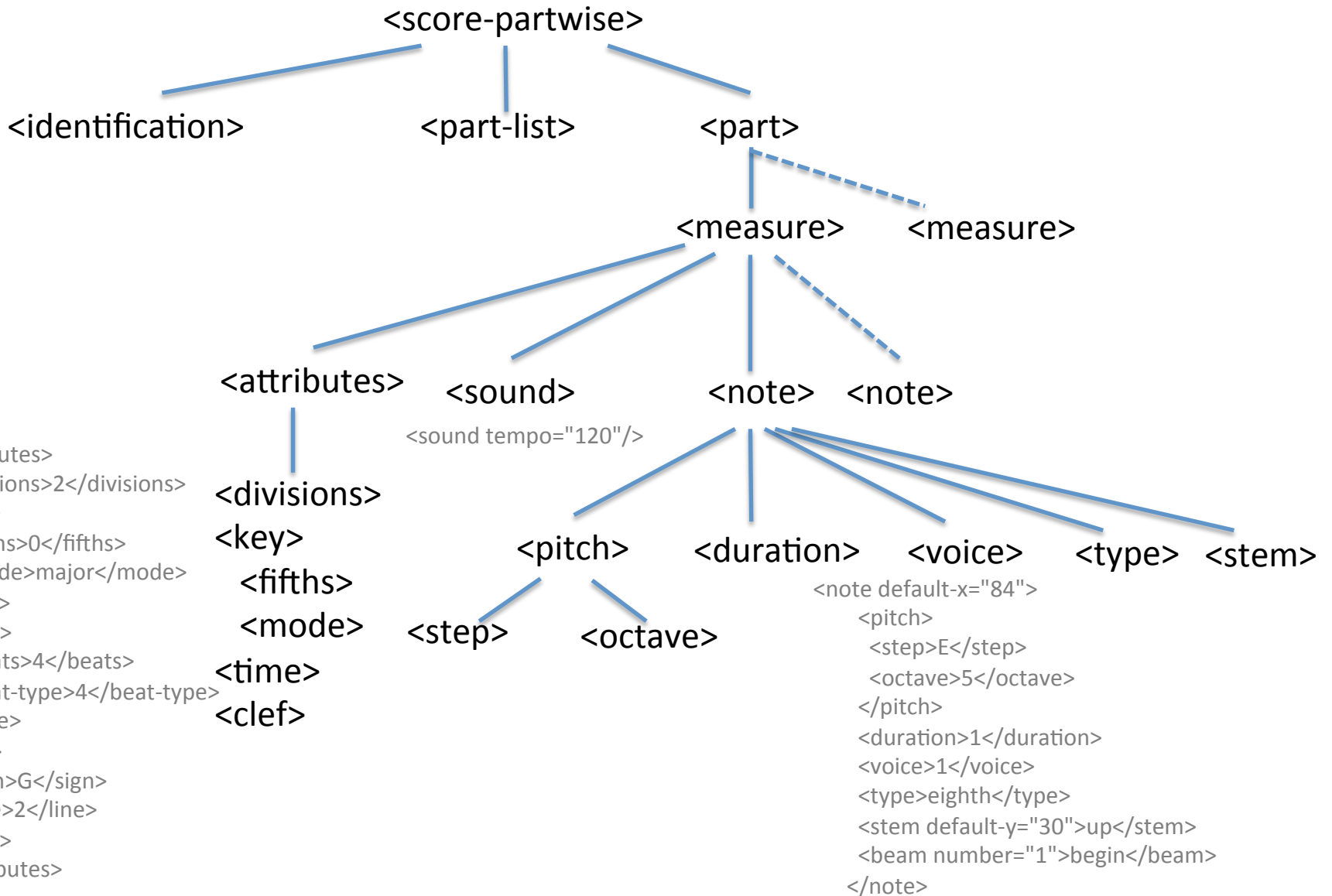
```
<identification>  
  <encoding>  
    <software>Finale 2012 for Mac</software>  
    <software>Dolet Light for Finale 2012</software>  
    <encoding-date>2013-02-25</encoding-date>  
  </encoding>  
</identification>
```

MusicXML Data hierarchy (header 2)

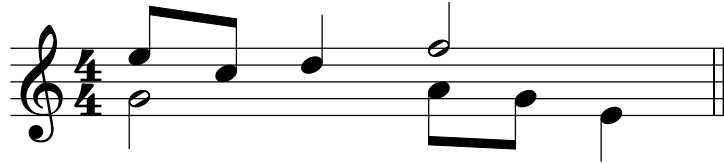


```
<part-list>
  <score-part id="P1">
    <part-name>MusicXML Part</part-name>
    <score-instrument id="P1-I1">
      <instrument-name>Garritan: ARIA Player</instrument-name>
    </score-instrument>
    <midi-instrument id="P1-I1">
      <midi-channel>1</midi-channel>
      <midi-bank>15489</midi-bank>
      <midi-program>1</midi-program>
    </midi-instrument>
  </score-part>
</part-list>
```

MusicXML Data hierarchy (part)



MusicXML Voices/Layers



MusicXML

MuseData

Voice 1:

```
<measure>
  <attributes>
    <note> E5, 1 tick
    <note> C5, 1 tick
    <note> D5, 2 ticks
    <note> F5, 4 ticks
    <backup> 8 ticks
```

Voice 2:

```
<note> G4, 4 ticks
<note> A4, 1 tick
<note> G4, 1 tick
<note> E4, 2 tick
</measure>
```

```
Group memberships: score
score: part 1 of 1
```

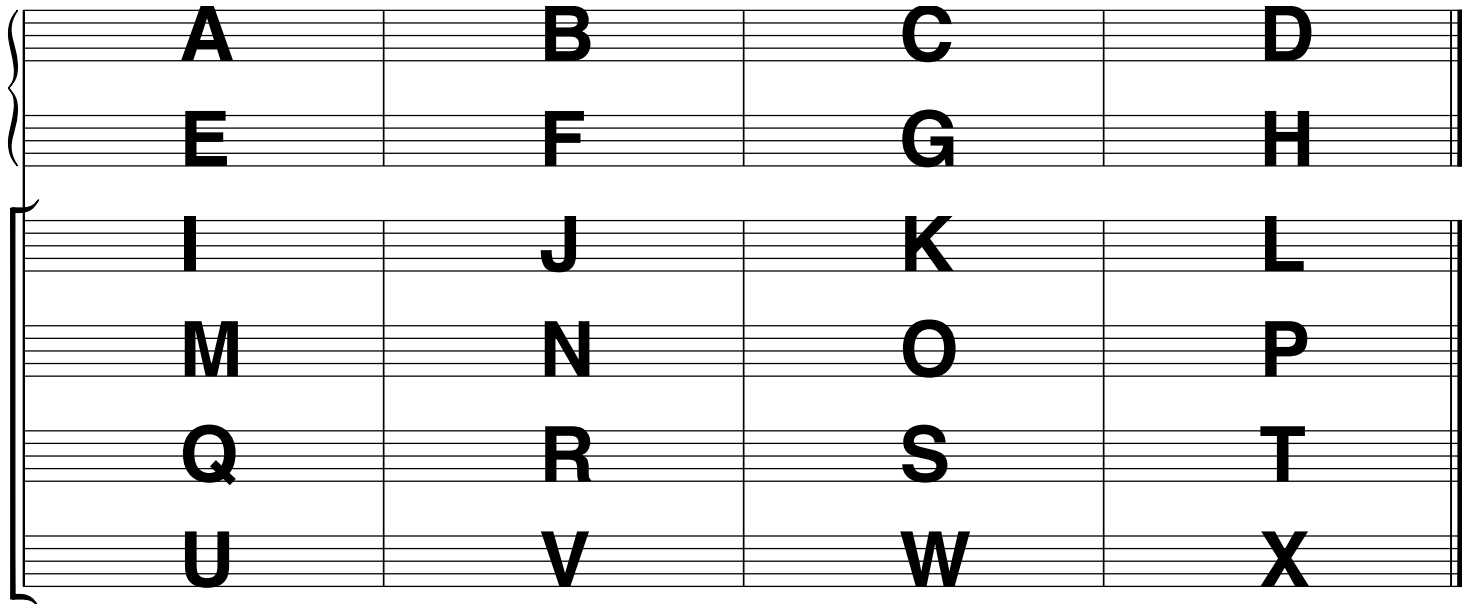
```
$ Q:2 K:0 T:1/1 C:4
```

```
E5      1      1 e      u  [
C5      1      1 e      u  ]
D5      2      1 q      u
F5      4      1 h      u
back    8
G4      4      2 h      d
A4      1      2 e      d  [
G4      1      2 e      d  ]
E4      2      2 q      d
```

```
mheavy2
/END
```

Partwise/timewise

- <score-partwise> stores score one part (staff) sequentially (part->measure)
- <score-timewise> score each measure sequentially for all parts (measure->part)
- <score-timewise> is about as common as MIDI Type-2 files.
- <score-timewise> is a quasi-realtime encoding (not strictly real-time).



- <score-partwise>: ABCD, EFGH, IJKL, MNOP, QRST, UVWX
- <score-timewise>: AEIMQU, BFJNRV, CGKOSW, DHLPTX
- <opus>: multiple movements of (partwise or timewise).

MusicXML versions

<http://www.musicxml.com>

<http://en.wikipedia.org/wiki/MusicXML>

MusicXML 1.0	2004
--------------	------

MusicXML 1.1	2005
--------------	------


MusicXML 2.0	2007
--------------	------

MusicXML 3.0	2011
--------------	------

MusicXML 3.0

- Compressed MusicXML: (.mxl): ZIP file which can include linked material as well as main XML file.
- Standardized list of instruments
 - <http://www.musicxml.com/dtds/3.0/sounds.xml>
 - <http://www.humdrum.org/Humdrum/guide.append2.html>
- Jianpu notation, microtonal music (Turkish music), AlphaNotes
 - <http://benny85erhu.wordpress.com/jianpu>
 - http://www.hinesmusic.com/What_Are_Makams.html
 - <http://blog.finalemusic.com/post/2011/10/20/Finale-Quick-Tips-AlphaNotes.aspx>
- More graphic symbol representations for percussion, handbells, haupt-, nebenstimme
 - <http://en.wikipedia.org/wiki/Hauptstimme>

Data Interchange Cases

Representation 1  Representation 2

