



Outline

- ◆ Introduction to MusicXML
- ◆ Design Issues
- ◆ Translation Issues
- ◆ MusicXML and Musical Analysis
 - XML Document Object Model (DOM)
 - XML Query (XQuery)



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

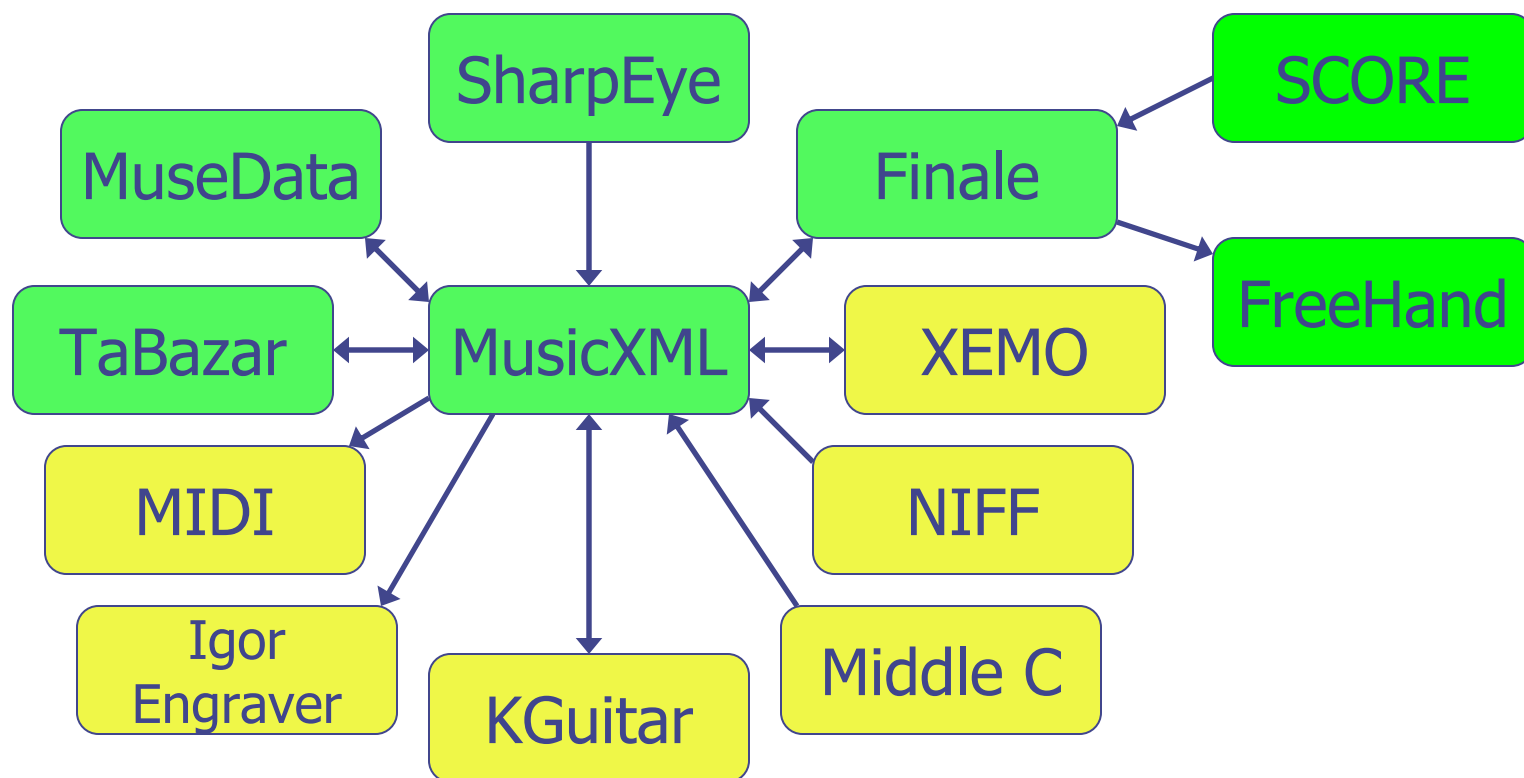


And It's Working

- ◆ MusicXML available under a royalty-free license modeled on W3C
- ◆ Supported by commercial programs: Finale, SharpEye Music Reader, Dolet
- ◆ Open source projects include Project XEMO and KGuitar
- ◆ Faster adoption than anything since MIDI



MusicXML Interchange Today





MusicXML Design Issues

- ◆ Logical domain in elements
- ◆ Visual and performance domains in attributes
- ◆ Each aspect of musical semantics represented in a different element
- ◆ Separate representations for what is heard vs. what is notated



in MusicXML (1 of 2)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
  "-//Recordare//DTD MusicXML 0.6b 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>
```

MusicXML vs. NIFF



Original as scanned into SharpEye



Imported into Finale via MusicXML



Imported into Sibelius via NIFF



Translation Issues

◆ MuseData

- Showed that XML works
- Duration problems in romantic repertoire

◆ NIFF and MIDI

- Can translate to/from NIFF's graphical format, but it's a lot of work
- Translating to MIDI straightforward
- Using XML as intermediate representation



Finale Translation Issues

- ◆ Lots of similarities, for instance between Finale layers and `<backup>/<forward>`
- ◆ What Finale structure becomes `<staccato>`?
 - A character that looks like a staccato dot?
 - A performance definition shortens the note length?
- ◆ Incomplete format documentation analagous to incomplete dictionaries



MusicXML Analysis Examples

- ◆ XQuery example
 - Simple melodic search
- ◆ Reporting vocal range
 - Needed for selling songs online
 - Implementation 1 with Visual Basic and XML Document Object Model
 - Implementation 2 with XQuery and QuiP



The Database Barrier

- ◆ Lack of an adequate, standardized music format and analysis tools inhibits building and sharing of musical databases
- ◆ Research tools that are available do not scale up to commercial use
- ◆ Enterprise-level databases are very costly: music needs to leverage other industries
- ◆ Relational database formats not well suited for music queries



XML Query Working Group

- ◆ "The mission of the XML Query Working Group is to provide flexible query facilities to extract data from real and virtual documents on the Web, therefore finally providing the needed interaction between the web world and the database world. Ultimately, collections of XML files will be accessed like databases."



XML Query Potential

- ◆ A computer industry standard that supports queries of MusicXML documents
- ◆ Over a dozen prototype implementations of the current XQuery working draft are available from:
 - Relational database companies like Oracle and Microsoft
 - Native XML database companies like Software AG
 - Other XML specialist companies



But Will XQuery Work for Music?

- ◆ Any old XML query language is not enough
- ◆ Problem scope needs to truly include documents as well as database tables
- ◆ Query language must support queries involving both sequence and hierarchy
- ◆ June 2001 Working Draft supported only sequence OR hierarchy
- ◆ Even trivial melodic search was very difficult



April 2002 Working Draft

- ◆ No support for full XPath 1.0 axes (ancestor, following-sibling, etc.)
 - Argument is that "turtle graphics" approach is impossible to optimize
- ◆ But there are new "<<" and ">>" operators
- ◆ Combined with supported XPath 2.0 features, looks adequate for music IR



Our First XQuery

- ◆ Search for all instances of C-D-E-C pitch step sequence in "Frere Jacques"
- ◆ Match across measures, but not across parts
- ◆ Use QuiP 2.1.1 prototype from Software AG

Frere Jacques Data



Soprano

Tenor

Bass

Musical score for the first system of Frere Jacques. It consists of three staves: Soprano, Tenor, and Bass. The Soprano staff begins with a treble clef and a common time signature. The Tenor and Bass staves begin with a bass clef and a common time signature. The Soprano part starts with a quarter note G4, followed by quarter notes A4, B4, A4, G4, F4, E4, and D4. The Tenor and Bass parts are silent for the first four measures, then enter in the fifth measure with a quarter note G2, followed by quarter notes A2, B2, and A2.

S

T

B

Musical score for the second system of Frere Jacques. It consists of three staves: Soprano (S), Tenor (T), and Bass (B). The Soprano staff begins with a treble clef and a common time signature. The Tenor and Bass staves begin with a bass clef and a common time signature. The Soprano part starts with a quarter note G4, followed by quarter notes A4, B4, A4, G4, F4, E4, and D4. The Tenor part starts with a quarter note G3, followed by quarter notes A3, B3, A3, G3, F3, E3, and D3. The Bass part starts with a quarter note G2, followed by quarter notes A2, B2, A2, G2, F2, E2, and D2.

S

T

B

Musical score for the third system of Frere Jacques. It consists of three staves: Soprano (S), Tenor (T), and Bass (B). The Soprano and Tenor staves are silent for the first two measures, then enter in the third measure with a quarter note G4, followed by quarter notes A4, B4, and A4. The Bass part starts with a quarter note G2, followed by quarter notes A2, B2, and A2.



Frere Jacques XQuery

```
<result>
{let $doc :=
  document("frere-jacques.xml")
let $notes := $doc//note
for $note1 in
  $notes[string(./pitch/step) = "C"],
  $note2 in $notes[. >> $note1][1],
  $note3 in $notes[. >> $note2][1],
  $note4 in $notes[. >> $note3][1]
let $meas1 := $note1/..
let $part1 := $meas1/..
let $part2 := $note2/../../..
let $part3 := $note3/../../..
let $part4 := $note4/../../..
```

```
  where string($note2/pitch/step) = "D"
    and string ($note3/pitch/step) = "E"
    and string($note4/pitch/step) = "C"
    and (string($part1/@id) =
      string ($part2/@id))
    and (string ($part2/@id) =
      string ($part3/@id))
    and (string ($part3/@id) =
      string ($part4/@id))
return
  <motif>
    {$note1/pitch} {$note2/pitch}
    {$note3/pitch} {$note4/pitch}
    <measure>{$meas1/@number}</measure>
    <part>{$part1/@id}</part>
  </motif>}
</result>
```



QuiP / XQuery Demo

The screenshot shows the Software AG QuiP application window. The title bar reads "Software AG QuiP". The menu bar includes "File", "Edit", "View", "Tools", and "Help". Below the menu bar is a toolbar with icons for file operations. The main window is divided into two panes. The left pane, titled "frere4a.xquery" and "frere4a2.xquery", contains the following XQuery code:

```
<result>
(let $doc := document("MusicXML/frere-jacques.xml")
let $notes := $doc//note
for $notel in $notes[string-value(./pitch/step) = "C"]
let $note2 := $notes[. >> $notel][1]
let $note3 := $notes[. >> $note2][1]
let $note4 := $notes[. >> $note3][1]
let $meas1 := $notel/..
let $part1 := $meas1/..
let $part2 := $note2/../../..
let $part3 := $note3/../../..
let $part4 := $note4/../../..
where string-value($note2/pitch/step) = "D" and
```

The right pane features the Software AG logo and the text "THE XML COMPANY". Below the logo, there are configuration options for the "Target":

- Tamino
- Server: localhost
- Database: myDb
- Filesystem
- Directory: "C:\Program Files\QuiP\..."

An "EXECUTE" button is located below these options. The bottom pane, titled "Output", displays the XML result of the query:

```
<?xml version="1.0"?>
- <quip:result xmlns:quip="http://namespaces.softwareag.com/tamino/quip/">
- <result>
- <motif>
- <pitch>
<step>C</step>
<octave>5</octave>
</pitch>
- <pitch>
<step>D</step>
<octave>5</octave>
</pitch>
```

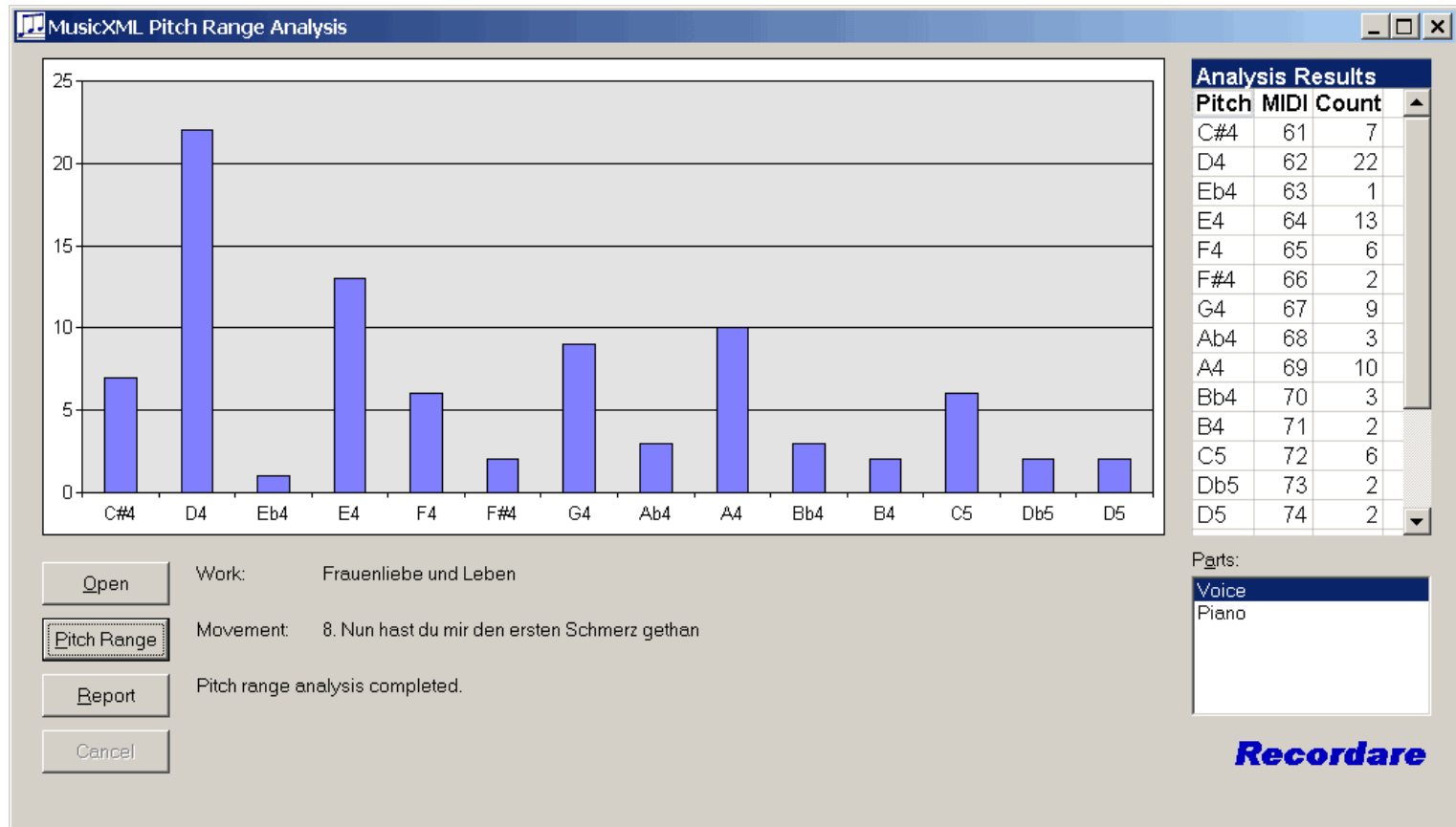
The status bar at the bottom of the window shows "Ln 1 Col 1" and "AUTO".



Pitch Range Analysis

- ◆ Want to know the lowest and highest notes in a vocal part
- ◆ Pitch comparisons can be done using MIDI value, but also want the note spelled correctly
- ◆ Want the first measure where the extremes occur for verification

Pitch Range Report in Visual Basic 6.0





Pitch Range XQuery (1 of 2)

```
define function MIDINote
  (element $thispitch)
  returns integer
{
  let $step := $thispitch/step
  let $alter :=
    if (empty($thispitch/alter)) then 0
    else if (string($thispitch/alter) =
      "1") then 1
    else if (string($thispitch/alter) =
      "-1") then -1
    else 0
  let $octave :=
    integer(string($thispitch/octave))
```

```
  let $pitchstep :=
    if (string($step) = "C") then 0
    else if (string($step) = "D") then 2
    else if (string($step) = "E") then 4
    else if (string($step) = "F") then 5
    else if (string($step) = "G") then 7
    else if (string($step) = "A") then 9
    else if (string($step) = "B") then 11
    else 0
  return 12 * ($octave + 1) +
    $pitchstep + $alter
}
```



Pitch Range XQuery (2 of 2)

```
let $doc :=
  document("/Frauenliebe8.xml")
let $part := $doc//part[./@id = "P1"]
let $highnote :=
  max(for $pitch in $part//pitch
      return MIDINote($pitch))
let $lownote :=
  min(for $pitch in $part//pitch
      return MIDINote($pitch))
let $highpitch :=
  $part//pitch[MIDINote(.) = $highnote]
let $lowpitch :=
  $part//pitch[MIDINote(.) = $lownote]
let $highmeas :=
  string($highpitch[1]/../../@number)
let $lowmeas :=
  string($lowpitch[1]/../../@number)
```

```
return
  <result>
    <low-note>{$lowpitch[1]}
    <measure>{$lowmeas}</measure>
  </low-note>
  <high-note>{$highpitch[1]}
  <measure>{$highmeas}</measure>
</high-note>
</result>
```




Conclusions

- ◆ MusicXML is the most widely adopted symbolic music format since MIDI
- ◆ Improving support for tablature, percussion, and sequencer applications
- ◆ Proven applications are a key part of standards process for both XQuery and MusicXML