μ-Script System : 楽曲データへのスクリプト埋め込みによる同期の実現

μ-Script System : Realization of Script-Embedding Synchronization on Music Data

阿部 秀彦　　五十嵐 健夫*

Summary. We propose a new framework to synchronize music and other elements in various applications . As computer processing speed gets faster and network technology improves, it has become simpler and more popular to create applications treating music contents. In those applications, synchronization with music is one of the most important matter. Previous systems, however, have not payed much attention. We designed a new system, μ-Script System, which makes possible the exact synchronization of application with music. This system is based on MusicXML and JavaScript. We implement this system as Java library using Rhino library, which includes interfaces with Java classes and Java objects.

1　はじめに

処理速度の向上や，ネットワーク技術の発展などによって，コンピュータによる音楽コンテンツが広く普及するようになってきた．例えばポップス曲にあわせてキャラクタが踊るアニメーションや，オーケストラの進行とともに打ちあがる花火，軽快な音楽にあわせて姿形を変える噴水などである．こういった音楽コンテンツにおいては，キャラクタの動き，花火の上がるタイミング，噴水が姿を変える瞬間などと，音楽の進行が同期していることは重要な要素である．

しかし，音楽の進行と同期するプログラムはリアルタイム性を必要とするため書くのは簡単ではない．音楽を手軽に扱う手法の一つとして，Macromedia Flash[1] や HTML 中の JavaScript などに見られるような，BGM などのリソースとして楽曲単位で扱う方法がある．この場合，同期は楽曲の演奏開始時にのみとられ，その後の楽曲中のタイミングは，あらかじめミリ秒やマイクロ秒といった時間単位で計算をして，起動タイミングを図るとことになる．この手法は理想的な環境ではうまくいく (図 1-a)．ところが一旦音楽の再生が遅れると，途中でプログラムとの同期が取られないため，それ以降ズレが続くこととなる (図 1-b)．また，計算機による音楽の再生は，どの計算機でも同じわけではなく，厳密には僅かではあるがズレが生じる．BGM としての音楽であれば多少のずれは充分許容されるが，音楽コンテンツの場合には，ミリ秒単位のズレが致命的となるため，初期化時の同期のみでは同期処理は不十分であるといえる．

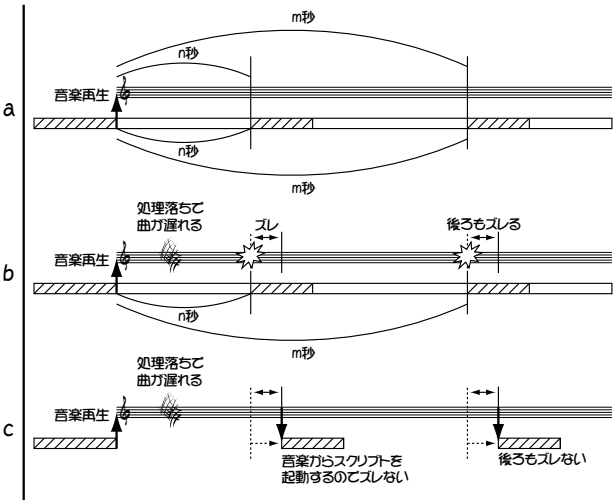

図 1. a:既存手法での音楽とプログラムの同期 (理想的な環境の場合)．b:既存手法での音楽とプログラムの同期 (処理落ちが生じた場合)．c:μ-Script System での音楽とプログラムの同期 (処理落ちが生じた場合)．

一方で，個々のコンテンツごとに特化した音楽プレーヤ内蔵のアプリケーションを実装するのはコストが非常にかかり，手軽に作成することはできない．さらに，一般にリアルタイム性を求めるプログラムの同期処理は複雑で深い知識が必要となる．

本稿では，音楽コンテンツ作成のための新しいシステム，μ-Script System を提案し，その Java ライブラリとしての実装について述べる．本システムでは，XML で表される楽曲中に JavaScript を埋め込み，XML を MIDI データに変換し，再生するシーケンサからプログラムを起動する．こうすることによって，プログラムは音楽との同期を常に保ち続け，不意のトラブルなどによる大きなズレや，ズレの蓄

* Hidehiko Abe, 東京大学大学院 情報理工学系研究科 コンピュータ科学専攻, Takeo Igarashi, 東京大学大学院 情報理工学系研究科 コンピュータ科学専攻 / 科学技術振興機構 さきがけ

積を抑えることができる (図 1-c)．加えて複雑な同期処理を隠蔽することで，アプリケーションの作成が単純で手軽になり，音楽コンテンツを作成するデザイナーなどにとっても有用である．

2 関連研究

スクリプトや XML を用いて時間ベースのコンテンツを作成する手法はこれまで幾つも提案されてきた．W3C によって提唱された SMIL(Synchronized Maltimedia Integration Language)[2] は XML 言語のひとつで，音声，音楽，ビデオ，アニメーション，テキスト，画像，などを用いてインタラクティブなプレゼンテーションを記述するための言語である．SMIL にはタイムラインの概念が導入されており，時間軸にあわせてイベントを起こすことが可能である．また，SMIL は HTML に似ているため，比較的容易に習得できる．

似たような言語に TIME(Timed Interactive Multimedia Extensions)[3] がある．これは HTML ドキュメント中に SMIL を埋め込むアプローチであり，Web ページに特化している．

しかしこれらの言語において，音楽は楽曲単位でリソースとして扱われ，時間はは秒やミリ秒などといった単位でのみ扱われる．このため，楽曲の開始と同期させて画像の表示，のようなことは行えるが，楽曲の途中にイベントを同期させることはできない．

音楽を楽曲単位だけではなく，その構成まで扱うスクリプトとして有名なものに CAL(Cakewalk Application Language) がある．CAL は MIDI の編集を行うアプリケーションのひとつである Cakewalk[4] 中で，MIDI イベントを処理するための lisp に似た関数型言語である．この CAL は MIDI のデータ作成中にその編集に用いられる言語であり，演奏中起動するようには設計されていない．

3 MusicXML

音楽とプログラムを同期させるためには，プログラムの起動タイミングを小節 · 拍などといった音楽中の位置で指定することが必要である．

指定方法の簡単な手法の一つは「何小節目の何拍目に何かをする」ということを記述した一覧を音楽と別に用意しておく方法である．しかしこの方法では楽曲のフレーズとプログラム起動タイミングとの関係が弱く，後から音楽データの方を挿入または削除など修正を行うと，編集したフレーズの部分だけでなく，直接は関係のない編集したフレーズ以降に対するプログラムの起動タイミングがずれてしまうなどといった問題が生じる．

そこで提案システムでは，音楽データ中に直接コードをスクリプトで記述するという方法をとることとした．これによって，音楽のフレーズに対応付けてプログラムの起動タイミングが指定できるため，音楽データの修正を行ったとき，編集したフレーズに伴って起動タイミングも前後し，直接関係のないフレーズに対応する部分に対する影響もない．

スクリプトコードを埋め込むために使用する音楽データフォーマットは MusicXML とした．MusicXML は Good[5] によって提唱された音楽情報を XML で表現したフォーマットであり，MIDI，mp3，wav，au，AIFF など，多くの音楽データフォーマットがバイナリ形式であるのに対し，MusicXML はテキスト形式である．Finale[6] や Sibelius[7] といった一般的な楽譜作成ソフトによってサポートされていることや，XML であるためインターネットとの相性のよさのために，近年急速に広まっている音楽ファイルフォーマットであり，近年活発に研究されるようになってきている．

MusicXML の規格には音楽演奏情報と楽譜記述情報の両方が含まれているが，本稿で提案するシステムでは特に演奏情報の方を使用する．

4 スクリプトの記述

スクリプトは本稿では JavaScript を用いることとし，`<script>...</script>`タグの中に文字列として記述できることとした．この手法は HTML 中に JavaScript を埋め込むときと同様である．

4.1 実行タイミングと記述位置

本システムでは，楽曲を再生する直前の初期化時に実行されるものと，楽曲再生中にイベントハンドラとして実行されるものの，2 種類のスクリプトを記述できることとした．

初期化時に実行するスクリプトは，`score-partwise` 要素の直接の子としての記述する．この `score-partwise` 要素は楽曲の根であり，すなわち各楽曲に対して唯一である．ここに記述されたスクリプトはインタープリタ初期化時に実行され，カウンターの初期化，関数の宣言などを行うことができる．

他方，イベントハンドラとしてのスクリプトの記述位置は `attributes` 要素の直接の子としての `script` 要素中である．MusicXML の中には `note`，`backup`，`forward` などのように長さの概念を持つ要素が存在する．これらの要素の中に記述してしまうと，その要素の開始時刻にスクリプトが実行されるべきか，終了時刻にスクリプトが実行されるべきか曖昧になってしまう．そのため，長さの概念をもたず，`note` 要素などと並列に並べられ，かつ記述される位置に対して楽曲中のタイミングが一意に定まる `attributes` 要素に `script` 要素を持たせることとした．変更 DTD を図 2 に示す．

例となる短いコードを図 3 に示す．

この例は，楽曲の開始から，2 つ目のスクリプトの位置までを 10 回繰り返すコードである．コード中

```
<!ELEMENT script (\#PCDATA)>
<!ELEMENT attributes (\%editorial;,
  divisions?, key?, time?, staves?,
  instruments?, clef*, staff-details*,
  transpose?, directive*,
  measure-style*, script?)>
<!ELEMENT score-partwise
  (\%score-header;, script?, part+)>
```

図 2. 変更した DTD

```
<score-partwise>
  ...
  <script>
    //初期化のスクリプト
    i = 0;
  </script>
  <part id="P1">
    <measure number="1">
      <!-- 音楽データ -->
      ...
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        ...
      </note>
      <note>
        <pitch>
          <step>D</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        ...
      </note>
      <note>
        <pitch>
          <step>E</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        ...
      </note>
      ...
      <attributes>
        <script>
          //ループのハンドラスクリプト
          i++;
          if( i != 10 ){
            //最初に戻る
            sequencer
              .setTickPosition(0);
          }
        </script>
      </attributes>
    </measure>
    ...
  </part>
  ...
</score-partwise>
```

図 3. サンプルコード

の“...”は，スクリプトと関係のない MusicXML のデータがあることを示している．また，スクリプト中で記号<はタグの開始記号と認識されてしまうため，使用したい場合はスクリプトを<![CDATA[...]]>の中に記述しなければならない．

4.2 外部アプリケーションとのインターフェース

本システムでは，より自由にアプリケーションが作成できるようにするため，ユーザーの作成した外部アプリケーションとの連携をとれるようにした．

ユーザーはシステムに対して，オブジェクトとその名前を渡すことで，スクリプト中で直接オブジェクトにアクセスすることができる．また，ユーザーがシステムに対して，クラスの情報とそのクラス名を渡すことで，スクリプト内部でそのクラスのインスタンスを生成することができる．例えば Java の System.out を"out"という名前で渡すと，スクリプト中に

```
out.println("μ-Script");
```

のように記述することで標準出力に文字列を出力することができる．また，ユーザープログラマが定義した独自のオブジェクトを渡すこともできる．

この機能を用いた小さな例を次にしめす (図 4). この例はカエルの画像が童謡カエルの歌にあわせて歌うアプリケーションである．

このアプリケーションに用いたスクリプトの最初の 1 小節分を，図 5 に示す．

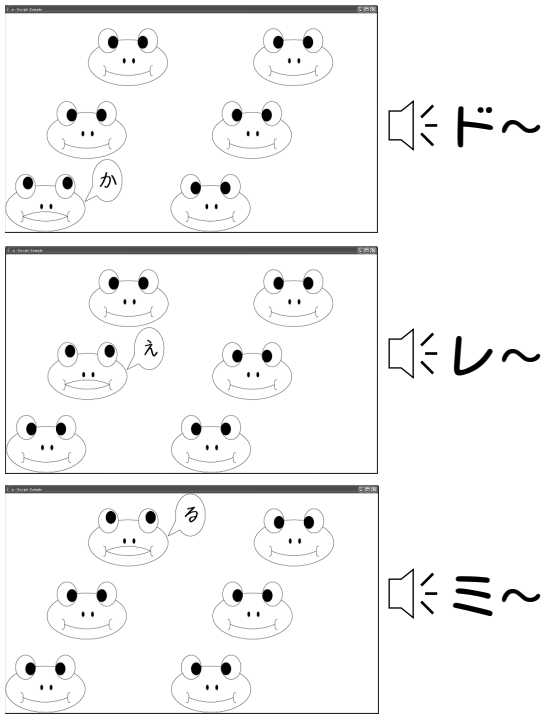

図 4. μ-Script System を用いたカエルの歌を歌うアプリケーション

```
<measure number="1">
  <attributes>
    <divisions>4</divisions>
    ...
    <script>
      DO.setMessage("か");
      frame.repaint();
    </script>
  </attributes>
  <note>
    <pitch>
      <step>C</step>
      <octave>4</octave>
    </pitch>
    <duration>4</duration>
  </note>
  <attributes>
    <script>
      DO.setMessage(null);
      RE.setMessage("え");
      frame.repaint();
    </script>
  </attributes>
  <note>
    <pitch>
      <step>D</step>
      <octave>4</octave>
    </pitch>
    <duration>4</duration>
  </note>
  <attributes>
    <script>
      RE.setMessage(null);
      MI.setMessage("る");
      frame.repaint();
    </script>
  </attributes>
  <note>
    <pitch>
      <step>E</step>
      <octave>4</octave>
    </pitch>
    <duration>4</duration>
  </note>
  <attributes>
    <script>
      MI.setMessage(null);
      FA.setMessage("の");
      frame.repaint();
    </script>
  </attributes>
  <note>
    <pitch>
      <step>F</step>
      <octave>4</octave>
    </pitch>
    <duration>4</duration>
  </note>
</measure>
```

図 5. カエルの歌の Sample Code

カエルの歌の歌いだし “かえるのうたが” に対応する音は “ドレミファミレド” である．このスクリプトはまず，最初の ‘ド’ が発音されると同時に，カエルの吹き出しに ‘か’ と表示する．次に，‘ド’ の発音が止まり ‘レ’ と発音されると同時に，‘か’ の表示を取り消し次のカエルにの吹き出しに ‘え’ と表示する．以下，同様に動き，カエルキャラクタが蛙の歌を歌っているように見えるようにスクリプトが埋め込んである．

このアプリケーションでは本システムに 7 つのオブジェクトが渡されている．Window のオブジェクトである `frame` オブジェクト，およびカエルに対応するオブジェクト，左から順に `DO`, `RE`, `MI`, `FA`, `SO`, `LA` である．

カエルオブジェクトには `setMessage` というメソッドが定義されており，このメソッドに文字列を渡し，`frame.repaint()` を呼び出すことで，対応するカエルの吹き出し中のメッセージが変更される．また，`setMessage` に `null` を渡すと，カエルは口を閉じ喋るのをやめる．

本システムはユーザーのプログラムから起動することとした．ユーザープログラム中でシステムのインスタンスを生成し，スクリプトの埋め込まれた XML ファイルを読み込み，`start` メソッドを呼ぶことで再生を開始する．簡単な例を以下に示す．

```
void main(){
    //インスタンスを生成
    MuScriptPlayer p
        = new MuScriptPlayer();

    // ファイルをロード
    p.load("***.xml");

    // 再生
    p.start();
}
```

またユーザーアプリケーションのオブジェクトをシステムに渡すには，

```
p.putInitObject("frame", f);
p.putInitObject("DO", frog_do);
p.putInitObject("RE", frog_re);
```

などのようにして名前とオブジェクトの参照を渡す．

5 実装

MusicXML には楽曲演奏情報は記述されているが，そのままでは演奏することができない．そのため本システムでは音楽演奏のために MusicXML を MIDI に変換することとした．

μ-Script System は大きく分けて 2 つの部分からなり，ひとつは音楽を再生するための MIDI[8] のシーケンサ，もうひとつはスクリプトのインタープリタである (図 6).

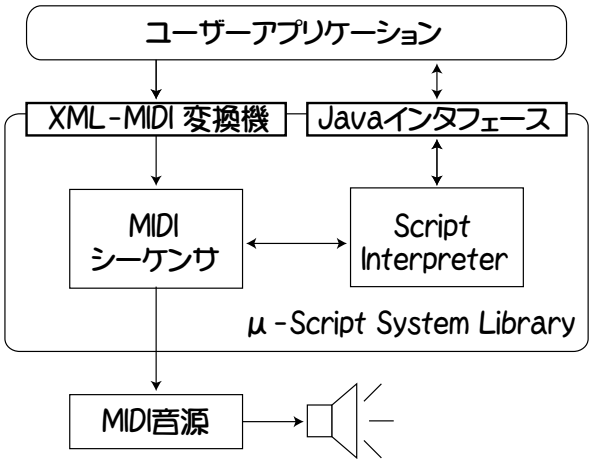

図 6. システム構成

MIDI シーケンサは時間を管理しながら MIDI イベントを発行し，スクリプトの記述されているタイミングに到達したとき，インタープリタを起動する．

インタープリタは対応するスクリプトを実行し，内部の状態の変更，外部とのコミュニケーション，シーケンサのコントロールなどを行う．

このライブラリを起動したときに行われる処理は以下の手順である．

1. スクリプト付 XML の読み込み
2. スクリプトインタープリタの初期化
3. MIDI への変換，スクリプトの事前コンパイル，スクリプト起動位置の保持
4. MIDI シーケンサの実行

5.1 MIDI シーケンサ

MIDI シーケンサがスクリプトインタープリタを起動するタイミングを検出するために，MIDI のメタイベントを使用することとした．メタイベントとは MIDI の規格において演奏以外のデータを記述したもので，特定のシーケンサとのコンタクトをとるためにも用いられるものである．メタイベントには様々な種類のものがあるが，その規格にはシーケンサ独自のデータを記述するための拡張可能な空きがあり，そのひとつを利用した．メタイベントは後ろに任意のバイト列を引数に取ることができるので，そこにスクリプトの ID を記しておくことで，記述されていたスクリプトを特定することとした．

5.2 スクリプトインタープリタ

スクリプトのインタープリタは，音楽のリアルタイム性を実現するために，出来る限り高速で動くものがよい．本システムでは，文字列を事前にコンパイルしておくことができる Rhino ライブラリ [9] を使用した．これによって実行時の速度を向上させることができる．

このライブラリには，Java との連携をとるためのインターフェースもそろっており，Java のオブジェクトを参照することや，新しいクラス定義を登録することができる．この機能を用いて，XML 中のスクリプトと外部 Java とのインターフェースを作成した．

さらにスクリプトの中で，現在のシーケンサの情報を得ることや，シーケンサを直接コントロールするために，インタープリタ初期化時に特殊なシーケンサオブジェクト `sequencer` が，インタープリタに渡される．このオブジェクトにアクセスすることによって，現在の再生位置の取得，変更や，再生テンポのコントロールなどを行える．

5.3 MIDI への変換

読み込んだスクリプト付の XML を MIDI に変換する手順を示す．MusicXML から MIDI への変換は，MusicXML のホームページにあるチュートリアルの中に指標が示されている．[10]

変換には大きく分けて，音符 · 休符などの長さの計算，ノートナンバーの計算，スクリプトの事前コンパイルの 3 つの手順が必要である．

MIDI と MusicXML では音符 · 休符などの長さの扱いが異なるため，そろえる必要がある．MIDI での長さは 1 拍を等分割した tick と呼ばれる単位長さの整数倍によって表される．tick の長さを表す 1 拍の分割数は解像度と呼ばれ 1 曲を通して固定である．他方 MusicXML での長さ `duration` も 1 拍を分割数 `division` で等分割したものを単位とする．しかし `division` は曲の途中で変更することができる．よって MIDI における解像度は現れる `division` の最小公倍数とした．

この解像度において，音符 · 休符などの長さは (解像度 $\times$ *duration*/現在の *division*) となる．しかし，この長さのとおり演奏すると音符と音符の間に切れ目が無くなってしまうため，発音する長さは計算された長さの 0.9 倍とした．また，0.9 倍した長さが整数となるように，解像度を 10 倍しておくこととした．

ノートナンバーの特定は，Good[11] によって紹介されている．以下の式を用いた．

$$n = (octave + 1) \times 12 + alter + \begin{cases} 0 & (\text{if } step = \text{‘C’}) \\ 2 & (\text{if } step = \text{‘D’}) \\ 4 & (\text{if } step = \text{‘E’}) \\ 5 & (\text{if } step = \text{‘F’}) \\ 7 & (\text{if } step = \text{‘G’}) \\ 9 & (\text{if } step = \text{‘A’}) \\ 11 & (\text{if } step = \text{‘B’}) \end{cases}$$

これらの計算によって，note を MIDI のイベントに変換することが出来る．音符は MIDI イベントとしては，発音 (Note on) と消音 (Note off) の 2 つのイベントに分割される．しかし，全ての音符をこの 2 つのイベントに分割するとタイの表現ができない．そこで，タイがつながっている場合，前の音符とつながっている場合は発音イベントを，後ろの音符とつながっている場合は消音イベントを，両方の音符とつながっている場合は両方のイベントを生成しないことで，タイを表現することとした．

スクリプトの事前コンパイルは，使用する Rhino ライブラリを用いて行う．事前コンパイルされたオブジェクトはシステム中に保存される．ここで，シーケンサの実装との対応を取るため，このコンパイルされたオブジェクトとインデックスとを対応付ける．また，このオブジェクトは，メタイベントの空きタイプのひとつである “8” 番，データにインデックスを入れた MIDI イベントに変換する．実行時には，そのタイミングで MIDI シーケンサがスクリプトを同期的に実行することとなる．

6 議論

本システムでは，音楽からスクリプトを起動するという手法をとった．この手法により，実行環境間の同期の取り方の差異を自然に吸収した形で，楽曲とスクリプトの同期をとることができる．

本稿の実装では，スクリプトインタープリタとして使用した rhino ライブラリの中で Java のシステムライブラリを用いることができ，スクリプト内に記述する部分と，Java によるユーザーアプリケーションに記述する部分の境界を，自由に決定することができる．例えば，4.2 で紹介したカエルの歌の例では，カエルに対応するオブジェクトを外部アプリケーション内で作成したが，μ-Script System 内で作成することも可能である．

こうした自由度は，例えば多人数でコンテンツを作成する場合は，デザイナーにも，その人のプログラミングの知識に応じた一部のコード記述してもらうことが可能となり，デザイナーの意図した演出をコンテンツ内に直接反映させやすくなる．一方，一人でコンテンツを作成する場合も，Script としてコーディングするか外部ユーザーアプリケーション内でコーディングするかを自由に選ぶことができるようになり，記述力を強化されている．ユーザーアプリケーションにとっては，“スクリプト付きの音楽” というリソース単位で扱えるため，同一のアプリケーションを用いて，複数のコンテンツを扱うことができる．これらの理由により，自由度が高いということは有用であると考えられる．

また，このシステムでのスクリプトの記述方法は他の，例えば HTML に JavaScript を埋め込むような一般的なスクリプトと同様の方法を採用し，さらに埋め込むスクリプトは簡単なイベントハンドラと同様なため，習得が容易であると考えられる．

7 まとめと今後の課題

本稿では，音楽との同期，特に音楽の途中と同期を取ることに主眼を置いたアプリケーションを簡単に作成するためのシステム μ-Script System を設計，提案した．また，このシステムの Java ライブラリを実装し，その内容を紹介した．このシステムを導入することによって，音楽と常に同期を取り続けるようなアプリケーションを作成するためのコストを下げることができると考えられる．

今後の課題としては，実世界へのデバイスとのインタフェースを直接取れるよう，また元となる XML を操作できるようライブラリを拡張したり，インターネットでの配信などに対応できるよう，ブラウザのプラグインの作成することなどが考えられる．両方を開発すれば，インターネットを介して，実世界のエンターテインメントデバイスを音楽にあわせて自由にコントロールすることが可能となる．

またこのスクリプトを簡単に記述できるような開発環境を提供することによって，より手軽にコンテンツを作成できるようになると考えられる．

参考文献

[1] Macromedia Flash MX, http://www.macromedia.com/.

[2] SMIL, http://www.w3.org/TR/REC-smil/.

[3] TIME, http://msdn.microsoft.com/library/default.asp?url=/workshop/author/behaviors/reference/time2/htime_node_entry.asp.

[4] Cakewalk. http://www.cakewalk.com/.

[5] M. Good. MusicXML: An Internet-Friendly Format for Sheet Music. In *XML 2001 Conference Proceedings*, 2001.

[6] FINALE, http://www.finalemusic.com/.

[7] Sibelius, http://www.sibelius.com/.

[8] MIDI, http://www.midi.org/.

[9] Rhino: JavaScript for Java, http://www.mozilla.org/rhino/.

[10] The MIDI-Compatible Part of MusicXML, http://www.recordare.com/xml/midi-compatible.html.

[11] M. Good. MusicXML in Practice: Issues in Translation and Analysis. In *Proceedings First International Conference MAX 2002: Musical Application Using XML*, pp. 47–54, 2002.