

# 音楽情報科学研究のための共通データフォーマットの確立を目指して

北原 鉄朗      橋田 光代      片寄 晴弘

科学技術振興機構 戦略的創造研究推進事業 CrestMuse プロジェクト /  
関西学院大学 理工学研究科

{t.kitahara, hashida, katayose}@ksc.kwansei.ac.jp

あらまし 本研究の目的は、音楽情報科学研究における様々な音楽データを統一的に扱う枠組みの構築である。音楽情報科学研究では様々な粒度の音楽データを扱うが、それらを統一的に扱う枠組みはなかった。我々が開発を進めている CrestMuseXML は、複数の XML フォーマットからなり、様々な粒度の音楽データを、リンクされたいくつかの XML ドキュメントとして表す。これにより、新たな XML フォーマットを追加することで容易に仕様を拡張できる。本稿では、CrestMuseXML およびそれに基づく音楽データを扱うための Java API の設計方針と現状の仕様について述べる。

## Toward Establishing Universal Data Format for Music Informatics

Tetsuro Kitahara      Mitsuyo Hashida      Haruhiro Katayose

The CrestMuse Project, CREST, JST /  
Graduate School of Science and Technology, Kwansei Gakuin University

**Abstract** The goal of this study is to build a unified framework for handling various music data used in music informatics research. Although various kinds of data are used in music informatics, no unified framework has been available for handling them. CrestMuseXML, which we have been developing, consists of several different XML formats; music data with different layers are described as different XML documents linked to one another. This makes it possible to extend the framework by adding a new XML format. This paper describes the design policy and current specifications of CrestMuseXML and the Java API for handling CrestMuseXML data.

### 1. はじめに

これまでに音楽情報科学に関する様々な研究が行われてきた。音楽情報科学の研究内容は、自動採譜や楽音の認識から、自動作曲・編曲、演奏の表情づけ、音楽情報検索、音楽表現用インターフェイスまで多岐に渡るが、そのほとんどは、音楽に関するあるデータを入力し、その音楽に関する異種の（多くの場合異なる粒度の）データを出力するタスクということができる。たとえば、自動採譜は音楽音響信号を入力としてその楽曲の楽譜に相当する情報を出力する問題であるし、演奏の表情づけでは、表情づけしたい楽曲の楽譜に相当する情報を入力し、その楽曲に対する演奏制御データを出力する。こうした音楽データの表現形式は、WAV 形式やスタンダード MIDI ファイル (SMF) などの一部のデファクトスタンダードを除き、それぞれの研究

者がそれぞれの事情に合わせて独自のものを定義して用いていた。そのため、各要素技術の開発が進んで高性能なものが出始めているにもかかわらず、それらを組み合わせて総合的な音楽情報処理システムを構築するのは困難であった。

近年の XML の普及に伴い、この状況は変わりつつある。ISO/IEC は、MPEG-7 と呼ばれる、音楽を含むマルチメディアコンテンツにメタデータを付与する XML ベースの規格を制定した<sup>1)</sup>。Good は、音楽の楽譜に相当する情報を記述する XML 規格 MusicXML<sup>2)</sup> を提唱し、様々なノーテーションソフトなどで広く用いられている。楽譜情報を記述するもう 1 つの主要な XML 規格としては、WEDELMUSIC Format<sup>3)</sup> がある。その他にも、Music Encoding Initiative (MEI)<sup>4)</sup> など様々なものが存在する。また、MPEG でも Symbolic Music Representation (SMR) の名で規格の標準化が進めら

れている<sup>5)</sup>。このように、XML をベースとした様々な音楽データフォーマットが登場し、複数システム間での相互利用の準備は整いつつあるように見える。

しかし、実際にはいくつかの問題が残されている。第1に、記述できる情報の種類が音楽情報科学研究用途としては不足している。音楽情報科学研究では、たとえば自動採譜における楽譜として整形する前の中間状態や、演奏分析・生成などにおける旋律の階層構造や演奏制御情報など、様々な種類のデータが考えられるが、これらのほとんどは、上で述べた XML 規格では扱われていない。第2に、上の述べた各々の規格は単一のレイヤー（粒度）のみを表し、異なるレイヤー間の関係性は記述できない。そのため、同一楽曲に対する演奏スタイルの違いの分析のように、異なるレイヤー間の関係性を分析するには、複数のフォーマットを使い分けた上で関係性を独自の方式で記述する必要が生じる。また、入力と出力で異なるレイヤーの音楽データを扱う場合には別々のフォーマットを用いることを余儀なくされるため、データ入出力部は複雑になる。第3の問題は、拡張性の欠如である。上述のように音楽情報科学研究で扱うデータの種類の多岐に渡るため、あらゆる場合を想定してあらゆる種類のデータフォーマットをあらかじめ用意するのは事実上不可能である。しかし、既存の規格で拡張性が十分なものは少ない。第4には、標準ツールの未整備があげられる。Proxy-Music<sup>6)</sup>、JMSL<sup>7)</sup> などいくつかの試みは存在するが、我々の目的を十分に達成できるものとは言い難い。

以上の状況を鑑み、CrestMuse プロジェクトでは、様々なレイヤーの音楽データを包括的に記述できる枠組みの構築を進めている。この枠組みでは、レイヤーごとに XML フォーマットを用意し、レイヤーの異なるデータは異なる XML フォーマットで記述する。レイヤーをまたがるデータ間の関係は XLink/XPointer を用いて XML ドキュメント間のリンクで表現する。これにより、個々のフォーマットをシンプルに抑えた上で幅広い種類のデータ記述をでき、拡張も可能になる。我々は、この方針の下で設計された複数の XML フォーマットを総称して CrestMuseXML と呼ぶ。また、Java 言語に基づく標準 API を提供することで、XML に詳しくない人でも容易にプログラムを書けるようにする。

音楽情報科学研究者間の交流を促進し分野全体の底上げを狙う研究は、すでにいくつか行われている。研究目的であれば自由に利用できる「RWC 研究用音楽データベース」<sup>8)</sup>、大量の商用楽曲に対するメタデータを利用できるようにする枠組みを提供する OMEN<sup>9)</sup>、共通のタスクに共通のデータセットで取り組んで切磋琢磨するコンテスト形式の MIREX<sup>10)</sup> などがある。こういった共通データベースはすでに研究コミュニティの活性化に一役買っているが、さらにデータフォーマット

を共通化することで、複数の音楽システムをシームレスにつなぎ、さらなる活性化を狙う。

本稿では、この CrestMuseXML とその API の開発に向けて関西学院大学片寄研究室が中心となって議論してきた内容をまとめ、現在開発中の API の設計方針を述べることにより、共通データフォーマットの確立に向けた議論の出発点を提供することを目的とする。

## 2. CrestMuseXML の基本方針

我々は、CrestMuseXML が備えるべき性質として、次の4つを要請する。

### 要請1 マルチレイヤー

粒度の異なる情報を統一的枠組みのなかで自然な形で記述し、処理できる必要がある。扱うべき内容としては、信号・波形レベルの情報から旋律や楽曲の構造、さらには感性情報など様々なものが考えられる。

### 要請2 既存規格との互換性

音楽データを表す XML 規格はすでにいろいろと存在し、利用され始めている。こうした利用実績のある規格を取り込むべきである。その際には、既存の規格を独自に拡張して用いるのではなく、既存の規格自体には変更を加えずに、その規格では扱っていなかった情報も一緒に扱えるようにすることが望ましい。既存規格への独自の拡張を行ってしまうと、その規格のバージョンアップ時に対応が難しくなると予想される。

### 要請3 拡張性

あらゆる種類のデータの記述方式をあらかじめ網羅的に定義するのは事実上不可能であるため、拡張性は必須である。個々の研究者が独自に拡張したものを矛盾なく統合できることが望ましい。

### 要請4 標準ツールの提供

XSLT や XLink といった周辺規格の充実により XML 自体が巨大な規格になりつつあるため、XML に精通していない人でも容易に利用できるようにする必要がある。そのため、ファイルの読み書きの標準 API や、この API を使った場合のプログラムの雛型、多くの研究者が共通に必要としそうな機能を実装したツールなどを提供する必要がある。

また、必須ではないが、次の点も考慮すべきと考える。

### 要請5 必要最小限度の複雑さ

標準規格を普及させていくには、学習の容易さも重要なポイントとなる。一般に、様々な応用に配慮して記述能力を高くするほど、規格が複雑になり、規格の習得に労力を要するようになる。そのため、幅広い応用可能性や高い記述能力を維持しながら複雑さを抑えることが重要である。

以上の要請に基づき、次の4つの基本方針を打ち立てた。方針1 レイヤーごとに XML フォーマットを設計

あらゆる種類の音楽データを単一の XML フォーマットで記述するのではなく、レイヤーごとに別々の XML フォーマットを設計し、異なる XML フォーマットの要素を XLink/XPointer でリンクする形をとる。この方法には、次にあげるメリットがある。第 1 に、同じデータにリンクしている別レイヤーの複数のデータを分析するといった応用が可能になる。たとえば同一楽曲の楽譜データにリンクしている様々な演奏家による演奏制御データを解析することで、演奏家の演奏の特徴を抽出するといったことである。第 2 に、レイヤーごとにデファクトスタンダードとなっているフォーマットがある場合には、それを変更なしに採用することができる。第 3 に、CrestMuseXML に含まれる複数の XML フォーマットのうち、自分の研究に関連するものだけを学習すればよいので、研究者の学習の労力を抑えることができる。

#### 方針 2 MusicXML を一切の規格変更なしに採用

我々が扱う音楽データのうち、楽譜に相当する情報は、MusicXML を用いて記述する。MusicXML は、ノテーションソフト Finale の他、様々なソフトウェアでサポートされている。また、文献 11), 12) などすでに MusicXML をベースとした音楽アノテーション研究が行われており、これらとの連携も考慮できる。楽曲の構造分析結果や演奏制御パラメータといった楽譜に立脚した情報の記述は、この MusicXML ドキュメントのノードを参照する形で記述する。

#### 方針 3 XML フォーマットの追加で仕様拡張を実現

CrestMuseXML を単一の XML フォーマットではなく複数の XML フォーマットの集合ととらえているので、CrestMuseXML の拡張は、CrestMuseXML の枠組み内での新たな XML フォーマットの追加という形で対応できる。既存のフォーマットに手を入れない限り、個々の研究者が独立に新たなフォーマットを追加しても競合は発生しない。

#### 方針 4 標準 API・ツールの開発に Java を採用

標準 API や標準ツールの開発には Java を採用する。Java はプラットフォーム非依存なプログラム言語として広く使われており、特に XML 関連のライブラリは非常に充実している。また、C++ などに比べて言語仕様がシンプルなので、言語の習得も容易である。さらに、MATLAB や Max/MSP でも Java のプログラムを利用できるとのことなので、これらの言語からの利用も期待できる。

以下で、この基本方針に則って開発中の CrestMuseXML およびその Java API について詳細を述べる。

### 3. CrestMuseXML の設計

#### 3.1 CrestMuse で扱う音楽データ

一般に、データは symbolic なものとそうでないもの

に分けることができる。楽譜は前者の代表で、音響波形は後者の代表である。しかし、あるデータが symbolic かどうかは必ずしも自明ではない。なぜなら、symbol というのは本来名義尺度であると考えられるが、symbolic な表現と呼ばれるものは、一部、間隔尺度などのより高い水準の尺度を含む場合があるからである（たとえば、楽譜における個々の音符の音の高さ）。そこで、CrestMuseXML では symbolic なデータとそうでないデータを特に区別せずに扱う。

代わりとなる考えとして、我々は扱うべき音楽データを notewise データ、イベントデータ、時系列データ、構造データの 4 種類に分けて考える。

- notewise データ

楽譜上の個々の音符に対するデータである。楽譜の音符が持つ情報自体（音の高さ、音価など）や演奏制御データ（その音符を弾くときに楽譜通りの時刻からどのくらいずらして弾くか、どのくらいの強さで弾くか、など）の他、音符ごとの音響的特徴も含む。音響的特徴としては、HTC<sup>13)</sup> で扱われている特徴量セットなどが考えられる。

- イベントデータ

時間軸上のある時点での出来事に関するデータである。代表例としては、ピアノのペダルのオン・オフがあげられる。ピアノのペダリングのように特定の楽器パートに対するものとテンポの変化のように全楽器パートに対するものとが存在する。

- 時系列データ

ある時間間隔（たとえば 10ms）ごとに、基本的には楽曲の最初から最後まで  $n$  次元ベクトルデータを並べたものである。これは、メル周波数ケプストラム係数 (MFCC) のような音響特徴量の他、音響信号自体も該当する。これらはデータ量が極めて大きいため、どのように記述するかはまだ未定である。そのため、時系列データは今後の課題とし、本稿では扱わないこととする。MPEG-7 のバイナリフォーマットなどを参考にして、効率的な記述方法を検討していきたい。

- 構造データ

複数のデータ間の関係を記述したものである。関係記述の対象となるデータは、上記 3 種類のデータである場合もあれば、それ自体が構造データである場合もある。MOMI<sup>11)</sup> で扱われている GTTM ベースの旋律のグルーピング構造などが該当する。

#### 3.2 CrestMuseXML の全体像

現在の CrestMuseXML の全体像を図 1 に示す。notewise データに関しては、MusicXML ドキュメントの note タグを XLink/XPointer で参照する形で記述する。イベントデータについては、どこかのデータにリンクを張るのではなく、何小節めの何拍めかを属性などに

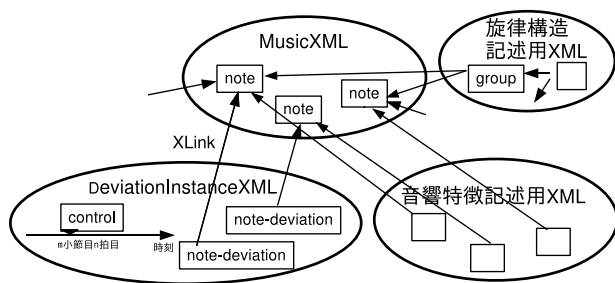


図1 CrestMuseXMLの全体像

記述することで、時間軸上の位置を指定する。構造データは、具体的な設計には至っていないが、MOMI<sup>11)</sup>に準拠する予定である。構造の記述の対象となるデータは、notewiseデータと同様にXLink/XPointerで指定することになる見込みである。

CrestMuseXMLに含まれる個々のフォーマットは、ほとんどが検討中あるいは検討開始前であり、仕様は確定していない。2008年8月に開催する演奏表情付けコンテストRenconで利用計画がすすんでいることから、2007年7月現在、演奏制御パラメータを記述するフォーマットDeviationInstanceXMLの仕様検討を先行して行っている。その他の記述の仕様検討は、これがある程度終わってから着手する予定である。

### 3.3 DeviationInstanceXML

Deviationとは本来「逸脱」という意味であるが、音楽情報処理（特に演奏表情付け）分野では、「音楽的な表現のための演奏における楽譜からの意図的な逸脱」という意味で用いられることがある<sup>14)</sup>。たとえば、同一の楽譜で表される楽曲を異なる演奏家が演奏するという場面で、その演奏に違いがあるとき、それらの演奏における楽器制御の仕方が、ある典型的な演奏のそれから逸脱しており、その逸脱の仕方の違いが演奏の違いであると考えられる。典型的な演奏としては、通常、いわゆる「表情なし」の演奏（発音時刻や消音時刻が完全に楽譜どおりで音の強さが全音一定）を取り上げる。

取り上げる楽器制御情報としては、設計の第1段階として、ピアノ演奏で用いられる制御情報を取り上げる。具体的には、各音符に対する発音時刻、消音時刻、音の強さ、テンポ、ペダルである。ピアノは、発音後は消音以外の音の制御が一切できず、発音や消音におけるパラメータも鍵盤を押したり離したりするときの強さだけであるので、比較的単純な仕様で見通しよく記述することができる。そのため、第1段階における検討材料としては適していると考えられる。

現行のDeviationInstanceXMLは、non-partwise, partwise, notewiseの3つのセクションに分けられる：  
`<deviation target="sample.xml"`  
`xmlns:xlink="http://www.w3.org/1999/xlink"`  
`<non-partwise>...</non-partwise>`  
`<partwise>...</partwise>`

```
<notewise>...</notewise>
</deviation>
```

non-partwiseは、イベントデータのうち全パートに共通なもので、テンポが該当する。テンポは、基準テンポを設定するtempoタグと実際のテンポを基準テンポに対する割合として記述するtempo-deviationの2つを用いる。tempo-deviationは「もたれ」などの拍ごとの細かなテンポ制御を記述するのが目的であるので、その効力はその拍だけとする。たとえば、

```
<non-partwise>
  <measure number="1">
    <control beat="1">
      <tempo>120</tempo>
    </control>
    <control beat="4">
      <tempo-deviation>0.8</tempo-deviation>
    </control>
  </non-partwise>
```

の場合、第1小節の1拍めで基準テンポが120に設定され、tempo-deviationがないので基準テンポ120のまま演奏される。4拍めにtempo-deviationがあるため4拍めはテンポが96(=120×0.8)となるが、次の拍からは120に戻る。

partwiseは、パートごとのイベントデータで、ピアノのペダリング、基準ダイナミクスの設定があげられる：

```
<partwise>
  <part id="P1">
    <measure number="1">
      <control beat="1">
        <base-dynamics>1.0</base-dynamics>
      </control>
      <control beat="4.06">
        <pedal action="on" depth="1.0"/>
      </control>
    </measure>
  </part>
</partwise>
```

各音符のダイナミクスは、個々の音符に付与されたダイナミクス値に基準ダイナミクス値をかけたものになる。これは、フォルテやピアノといった大局的な指示に対応したダイナミクスは後者で記述し、前者はフレーズ内の細かなダイナミクス制御を記述することを意図している。ペダリングは、MusicXMLにも記述する方法が用意されているが、演奏制御情報としてこちらで用意している。MusicXMLドキュメント内のペダル情報はあくまで楽譜の指示であり、実際の演奏におけるペダリングとは必ずしも対応しないからである。

また、上の例のように、beat属性に実数を指定することで、細かなイベント時刻を指定できる。

notewiseは文字どおり各音符に対するdeviation情報であり、以下のように記述する：

```
<notewise>
  <note-deviation
    xlink:href="#xpointer(//part[@id='P1']/
```

```

        measure[@number='1']/note[2])">
<attack>0.1</attack>
<release>-0.1</release>
<dynamics>0.7</dynamics>
<end-dynamics>0.7</end-dynamics>
</note-deviation>
        .....
</notewise>

```

これは、P1 という ID が付与されたパートの第 1 小節の 2 つめの note 要素に対する deviation 情報である。attack と release は楽譜通りの発音時刻・消音時刻からのずれを表し、1 拍を 1.0 とした実数で、後ろにずれる場合に正の数、前にずれる場合に負の数を使って記述する。dynamics と end-dynamics は発音時・消音時の強さで、後に述べる基準ダイナミクスに対する割合として記述する。

### 3.4 MIDI XML

MIDI XML は、Recordare が配布している MusicXML の DTD 群に付属している規格で、SMF をそのまま XML 化したものと考えればよい。MIDI XML 自体は Recordare 独自のものと思われるが、SMF に含まれる個々の MIDI イベントの記述は、MIDI Manufacturers Association が公開している MIDI XML Specifications を用いており、標準規格として扱って差し支えないと思われる。そこで、CrestMuseXML では、SMF と 1 対 1 で対応する XML フォーマットとして MIDI XML を採用する。ただし、MIDI XML は 3.1~3.2 節で述べた考え方・方針とは完全には一致しない。

## 4. Java API の設計

API 提供の目的は、様々な応用で共通な処理の流れや機能をあらかじめ実装しておくことで、個々の音楽システム的设计者の負担を軽減することにある。音楽システムは、インタラクティブシステムとバッチ型システムに大別できる。後者は多くの場合、指定されたファイルを読み、そこに含まれる音楽データに対して何らかの処理を行い、処理結果をファイルに書き出すという流れになる。そこで、現在のバージョンではバッチ型システムを念頭において、この一連の流れの雛型を提供する。これを次の基本方針で実現する。

- 個々のバッチ型システム（コマンドと呼ぶ）の共通基底クラスを用意し、コマンド作成で共通に必要な機能を提供する。
- 個々の XML ドキュメントやドキュメント内の個々の要素をラップするクラスを用意し、ドキュメントや要素へのアクセスをこれを通じて行う。

以下、設計の詳細を述べる。

### 4.1 コマンドの基底クラス

コマンドの基底クラス CMXCommand が用意されており、コマンドはこれを継承して作成する（図 2）。

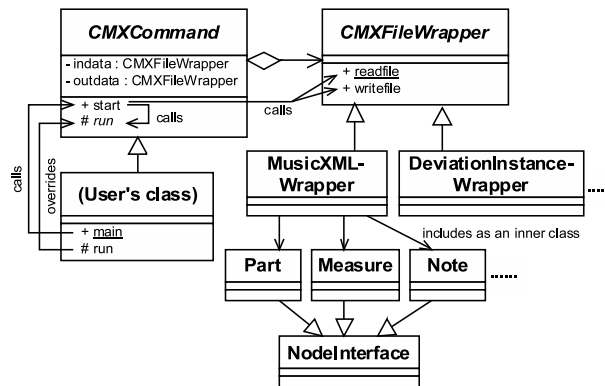


図 2 本 API におけるおおまかなクラス図。

継承する際に、抽象メソッドとして宣言されている run をメソッドオーバーライドし、ファイル読み込み後の処理の内容を記述する。そうすれば、start メソッドを呼び出すだけで、上記の一連の処理を自動的に行う。また、オプションを処理する仕組みも提供する。

### 4.2 ファイルラップとノードインターフェイス

XML ドキュメントをラップするクラスをファイルラップと呼び、基底クラス CMXFileWrapper のサブクラスとして提供される。たとえば、MusicXML ドキュメントは MusicXMLWrapper クラスがラップする。XML ドキュメント内の個々の要素をラップするクラスは、NodeInterface クラスのサブクラスとして提供される。たとえば、MusicXML ドキュメントの part, measure, note 要素をラップするクラスとして、Part, Measure, Note クラスが MusicXMLWrapper の内部クラスとして提供される。各要素から情報を取り出すのは、これらのクラスを通じて行う。ただし、単純さを保つため、NodeInterface クラスはデータの取得のみサポートし、設定はサポートしない（immutable）。

### 4.3 二分木を用いたノートビュー

ある音符に対して、その音符に閉じた情報（音の高さや長さなど）が欲しいだけであれば Note クラス内のメソッドだけで行えるが、次に来る音符や同時に鳴る他の音符の情報が欲しい場合には、それだけでは行えない。そこで、楽譜上の音符のならばを図 3 のような二分木に見立て、二分木上をたどることで、上のようなことを実現する。このように音符の集合を二分木に収めたものをノートビューと呼ぶ。ノートビューは用途に合わせて複数のものが用意される。たとえば、全パートの音符をすべて 1 つの二分木に収めたものやパートごとに別々の二分木におさめたものなどである。前者は、ある音符と同時に鳴る音符を他のパートも含めて取得する場合、後者は、ある音符に対してパート内での次の音符を取得する場合などに有用である。スラーもノートビューを用いて管理する。あるスラーに対してスラーがかかった音符の集合を取得したり、ある音符に対してその音符を含むスラーを取得したりという

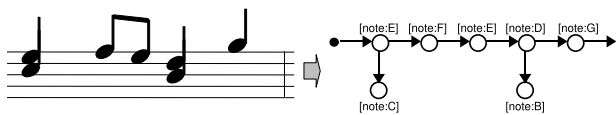


図3 二分木を用いたノートビュー。この二分木では、右の子へのパスが音符間の時間的な関係を表し、左へのパスは同時に発音する音符（順不同）を表す。

ことがノートビューを介してできる。

#### 4.4 ドキュメント走査の共通化

XML フォーマットごとの典型的なドキュメント走査手順を各ラッパクラスに実装する。たとえば、MusicXML ドキュメント（ただし partwise）は part 要素の子として measure 要素があり、その子として note 要素があるという木構造をなす。この木構造を行き帰りがけ順<sup>\*</sup>で走査する。この木の走査は MusicXMLWrapper クラス内で実装し、走査中に行う処理内容（タスク依存）はイベント駆動の考え方をを用いて別クラスに実装する。走査メソッドは part, measure, note の各要素をたどるときにイベントを発生し、指定されたハンドラがそれを受けて必要な処理を行う。ハンドラは、beginPart, endPart, beginMeasure, endMeasure, processMusicData の各メソッドが宣言されたインターフェイスを実装したクラスとして実現される。

### 5. 開発の現状と今後の予定

以上の設計に基づき、現在、CrestMuseXML API を関西学院大学片寄研究室内で開発中である。北原と橋田が仕様について議論した上で、北原がCrestMuseXMLの仕様を策定し、Java API を開発している。橋田および片寄研学生がAPIを自身の研究で試用し動作テストを行っている。片寄研内部サーバ上で作動しているsubversionでバージョン管理しており、片寄研内での共同開発ができるようになっている。機能拡充を片寄研で分担しながら開発を進めていく予定である。

ただし、フォーマットの策定およびAPIの開発は片寄研究室に閉じて行われるものではなく、CrestMuseプロジェクトメンバー、さらに国内外の音楽情報科学研究者と共同で進めていくことを想定している。まずは、仕様の要望を広く募集して我々が開発に反映させるところからはじめ、ゆくゆくは公開サーバ上にsubversionレポジトリを用意して、不特定多数の有志と共同で開発する環境を整えていきたい。第1歩として、CrestMuseXMLの開発に関する議論をするためのメーリングリストを作成したので、興味のある方はご参加いただきたい。また、CrestMuseXMLの最新情報は<http://www.crestmuse.jp/cmxml/> で公開する予定である。

前述のように、CrestMuseXMLは2008年8月の

<sup>\*</sup> 行きがけと帰りがけの両方で要素をたどることを表す、本稿での造語。葉以外の要素は2回たどることになる。

Renconでの利用が予定されている。そのため、CrestMuseXMLのうちRenconにかかわる部分が、Rencon Kitの一部として先行配布される見込みである。

### 6. おわりに

本稿では、音楽情報科学研究で用いる様々な種類のデータを共通の枠組みで扱うための試みとして、CrestMuseXMLとそのJava APIについて述べた。CrestMuseXMLを複数のXMLフォーマットの集合として定義することで拡張性を担保し、Java APIでは、異なるXMLフォーマットを共通の枠組みで扱えるようにして、バッチ型システムの雛型を提供することで、バッチ型の音楽システムであれば、見通し良く実装できるようにした。

しかしながら、様々な音楽情報科学研究者にとって有用なものになるには、多くのものが不足している。まず、今回扱わなかった様々な音楽データに対してXMLフォーマットを策定する必要がある。それに合わせ、APIを拡張する必要もある。プログラミング初心者でも使えるように、チュートリアルなどの文書も必要であろう。また、リアルタイム型システムへの対応をどう行うかも未定である。これらをすべて我々の力だけで行うのは困難である。ぜひ、この試みを成功させるためにご協力いただければ幸いである。

謝辞 ご討論いただいた長田典子氏（関西学院大学）、野池賢二氏に感謝する。

#### 参考文献

- 1) Manjunath, B. S., Salembier, P. and Sikora, T.: *Introduction of MPEG-7*, John Wiley & Sons Ltd. (2002).
- 2) Good, M.: MusicXML: An Internet-Friendly Format for Sheet Music, *XML 2001 Conf. Proc.* (2001).
- 3) Bellini, P. and Nesi, P.: WEDELMUSIC Format: An XML Music Notation Format for Emerging Applications, *Proc. WEDELMUSIC*, pp.79-86 (2001).
- 4) Roland, P.: The Music Encoding Initiative (MEI), *Proc. MAX 2002*, 2002.
- 5) Billini, B. and Nesi, P.: Symbolic Music Representation in MPEG, *IEEE Multimedia*, pp.42-49 (Oct.-Dec. 2005).
- 6) <https://proxymusic.dev.java.net/>
- 7) <http://www.algomusic.com/jmsl/>
- 8) 後藤他: RWC 研究用音楽データベース: 研究目的で利用可能な著作権処理済み楽曲・楽器音データベース, *情処学論*, **45**, 3, pp.728-738 (2004).
- 9) D. McEnnis *et al.*: Overview of OMEN, *Proc. ISMIR*, pp.7-12 (2006).
- 10) J. S. Downie *et al.*: The 2005 Music Information Retrieval Evaluation Exchange (MIREX 2005): Preliminary Overview, *Proc. ISMIR*, pp.320-323 (2005).
- 11) 平田他: MOMI: 音楽メタ情報記述のためのフレームワーク, *情処研報*, 2005-MUS-61, pp.13-17 (2005).
- 12) 梶, 長尾: 楽曲に対する多様な解釈を扱う音楽アノテーションシステム, *情処学論*, **48**, 1, pp.258-273 (2007).
- 13) 亀岡他: 調波時間構造化クラスタリング (HTC) による音楽音響特徴量の同時推定, *情処研報*, 2005-MUS-61, pp.71-78 (2005).
- 14) 豊田他: 音楽解釈研究のための演奏 deviation データベースの作成, *情処研報*, 2003-MUS-51, pp.65-70 (2003).