통합 방식을 이용한 대보표 악보의 XML 스키마 모델

김정희* · 곽호영**

XML Schema Model of Great Staff Music Score using the Integration Method

Jeong-Hee Kim* · Ho-Young Kwak**

요 약

현재 악보에 대한 DTD(Document Type Definition) 정의는 응용에 따라 그 종류가 다양하게 연구되어 왔으며, 또한 정의된 DTD를 XML Schema로 자동 변환하는 방법들이 진행되고 있다. 그리고 연구되어온 DTD 정의 구조는 악보 상에 존재하는 음악 정보들이 각각 독립된 형식을 유지하도록 표현하는 데 집중되고 있다. 따라서, 본 논문에서는 악보의 기본 구성 요소가 소절(Measure:마디)인 점을 이용하여 소절 단위 내 관련된 정보를 일련의 연속 문자열 값으로 표현하는 방법을 제안하고 이를 위한 XML Schema를 모델링 하였다. 또한 제안된 방법에 의해 표현된 XML 인스턴스에서 음악 정보를 추출하는 메카니즘을 제시하였다. 그 결과 연속된 문자열 값들을 가지는 XML Schema 정의가 가능했으며, 제안된 방식을 적용한 인스턴스는 기존 방식의 인스턴스들 보다 검색식(XPATH) 표현의 단순화 및 검색 단계를 줄여 그 효율성이 기대된다. 또한 사람에 의한 직접 표현 능력이 가능하며, 인스턴스 크기가 작아짐을 알 수 있었다.

ABSTRACT

Currently, DTD(Document Type Definition) Definition of Music score has been widely studied according to applications, and the methods of automatic transformation from defined DTD to XML Schema is in progress. In addition, studies of structure of DTD definition are focused on the the expression of music information by individual format. In this paper, expression method of the music information by continuous string values is suggested using the fact that measure is basically a component of score, and XML Schema is also modelled. In addition, mechanism extracting the music information from XML instance which was expressed using the proposed method is presented. As a result, XML Schema taking the continuous string values could be defined, instance obtained by the proposed method results in increasing efficiency by simplicity of XPATH and reduction of search step compared to previous method. In addition, it is possible for human to make direct expression, and it is known that the instance size decreases.

키워드

XML, DTD, SCHEMA, XPATH, MEASURE

Ⅰ. 서 론

오늘날 웹과 네트웍의 발달로 인하여 많은 멀티미디어 정보들이 쏟아지고 있고 이에 따라 멀티미디어 정보에 대한 검색 시스템이 활발히 연구중에 있다. 아직까지 음악정보 검색에 대한 연구는 미비한데 최근 차세대 웹문서의 표준인 XML(eXtensible Markup Language)[2]이 인터넷상에서 정보교환의

*제주대학교 대학원 박사과정

**제주대학교 통신컴퓨터공학부 교수

접수일자 : 2002. 12. 11

표준으로 자리 잡으면서 서양음악을 표현하기 위한 방법으로서 MusicXML, ScoreML과 같은 포맷이 고안되었다[1].

이와 같이 현재 XML언어는 인터넷 Web 문서, 전자 도서관, CSCW(Computer Supported Cooperative Work), CALS(Commerce At the Light Speed), 수학 분야의 MathML, 화학식 표현, 또는 응용프로그램간의 데이터 이동, 클라이언트와 서버간의 정보 이동 등에 적용되고 있으며, 채널기술의 CDF(Channel Definition Format). PDA(Personal Digital Assistant) 분야의 데이터 표현 등에서 상당히 활발한 연구들이 진행[3,4,5,6]되고 있고 있지만 기존의 음악을 디지털로 표현하는 기술은 음악의 복잡성으로 인하여 Finale, NIFF, SMDL, MIDI 등 여러 포맷들이 사용되어지고 있으며, Finale는 상업용으로서 표준이 되기가 힘들고, NIFF, SMDL은 모두 사람 중심이 아닌 기계 중심으로 설계되어 있어서 애플리케이션들이 이를 이용하기가 힘들며, 또한 MIDI는 전자음악 산업에 지대한 공을 세운 포맷이지만 표기법, 분석, 검색에 한계가 있다[1]. 따라서, 본 논문은 XML 스키마를 이용하여 음악적 구성 요소들을 기존의 독립화된 표현 기법과는 달리 음악 구성 요소의 최소단위인 소절단위[7]로 그룹화하면서, 사람이 직접 표현할 수 있는 표현법으로 설계하고 모델링하며 표현된 방식을 분석하는 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서 기존 관련 연구를 살펴보고 3장에서는 필요한 구성 요소들과 전체 스키마를 모델링한다. 4장에서는 모델링 검증 및 알고리즘 제시, 그리고 5장에서는 결론 및 향후 연구 방향을 기술한다.

II. 관련 연구

1. SMDL

마크업 언어이면서 SMDL(Standard Music Description Language)은 ISO(International Standardization Organization)에서 정의된 HyTime(Hypermedia/Time-based Structuring Language)과 SGML의 응용이고, 그 기능은 여러 가지 음악 작업 영역(Domain)들을 표현하기 위한 수단이며, 그 영역이 4가지로 Logical Domain, Visual Domain, Gestural and Analytical Domain으로 구분된다. 즉, 기본적으로 악보 표현과 특히 성가곡인 경우에는 사람의 몸짓까지도 표현하도록 한다. 또한 HyTime의 Link 기능을 이용하여 악전 표현의 비트맵 기호를 이에 해당되는 또 다른 마크업 언어로 변환하는 응용프로그램 간 그 표현이 서로 교환이 되도록 인터페이스 지원도 가능하다[8].

2. MNML

MNML(Musical Notation Markup Language)은 악보상의 기호와 가사들을 가능한 한 간결하게 표현하기 위하여 구문론적으로 기술된 마크업 언어이다. 음표를 표현하기 위한 예를 보면, 옥타브는 사람이 들을 수 있는 가장 낮은 "C"음을 "0" 옥타브로 두면서 3옥타브(middle C)까지 표현하며, 음표 값을 "A"에서 "G"까지 표현하는데 이는 음표의 기둥이 위로 향하는 것을 기본으로 한다. 또한 "a"에서 "g"까지는 음표의 기둥이 밑으로 향할 때 사용한다. 반음 올림은 "#", 한음 올림은 "x", 반음 내림은 "-", 한음 내림은 "=", 기본음은 "n"을 사용한다. 표 1은 MNML을 사용한 표현들이다.

하지만 MNML은 꾸밈음에 대한 표현이 지원되지 않는다[9].

표 1. MNML 표현
Table. 1 Expression of MNML

MNML 표현	설 명
<1:3en:4:40>	첫째 보표에 3옥타브 기본음 "미"음의 4분음표를 이전 세로줄 오른쪽에 40pixel의 공간을 두고 표현
<2:2C#:5:80:h1,3><2:D::100:h1,3><2:E::120:h1,3>	3개가 하나의 대상을 표현한 것인데, 먼저, (h1,3)은 보(beam:h) 차례가 1이면서 3개의 음표(8분음표)가 그룹화 됨을 표현했고, 그리고 두 번째 보표이고, 음은 C#,D,E이면서 기둥은 아래 방향임을 표현함.

3. MHTML

MHTML(Music HTML)은 HTML의 표현법을 악보 표현에 응용하고 확장시킨 기호법이다. 예를 들면 중첩된 슬러(Slurs)들을 표시하기 위해 종료 태그에 속성들을 사용하는 기법을 쓰며, 매우 직관적인 표현 특성을 갖고 있다. 하지만 XML의 기준을 따르

지 않고 있으며, 따라서 표준 파서(Parser)에 의해 분석되지 못한다. 단지 HTML의 확장 표현이다[10]. 그림 1은 MHTML로 표현하기 위한 악보이고 표 2는 이를 MHTML로 표현한 것이다.

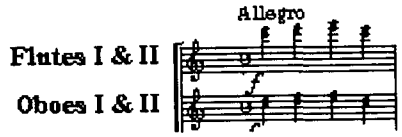

그림 1. 예제 악보
Fig. 1 Example Sheet

표 2. MHTML 표현
Table. 2 Expression of MHTML

```
<staff text = "Flutes I & II" next = "Fl. I & II">
 <parts onestem = beam
        same = stems samemeasure = a2>
 <part text = I>  <copy name = I>
    <tempo text = "Allegro" 4 = 132
           over = global over = "staves:strings">
    <dynamics f>
  <base octave = 5>e f g e <bar></base>
 </copy> </part></parts>
</staff>   ..... 이하 생략
```

4. MML(Music Markup Language)

SGML의 부분집합으로 악보를 HTML과 XML형식으로 웹상에서 볼 수 있도록 지원한다. 특히 이 언어는 음악을 만들어 내는데 있어서 연주 형식이나, 악기의 편성, 또는 음악을 다룰 때 특별한 형식 등을 기술할 수 있도록 지원하고 있지만 사용할 수 있는 모든 속성들에 대한 정의가 현재도 진행 중에 있으며, 표 3는 바흐(Bach)의 "Now that the sun doth shine no more" 곡인 그림 2상의 박스(Box) 부분을 MML로 표현한 것이다. 그리고 표 4는 MML의 표현 영역인 모듈들이다[11].

표 3. MML 표현
Table. 3 Expression of MML

```
<upbeat id="treb">[A E] </upbeat>
<bar barid="treb-1">3[E A] [3E B] [3E:8 C]F:8
[3G C]
    <notation>
    <bindbegin id="b1" beat="3" note="E" />
    <bindend end="b1" beat="3.5" />
    </notation>
</bar>
```

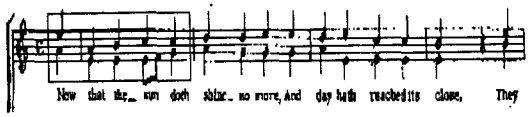

그림 2. 예제 악보
Fig 2. Example Sheet

표 4. MML 모듈
Table. 4 Modules of MML

모 듈	구성 요소
Frequency	Cent, Note, Octave, Scale, Turnig 등
Time	Note, Rhythm, Length, Synchronization 등
Organization	Album, Playlist, Head, Title, Song, Phrase, Link 등
Texture	Intensity, Envelope, Harmonics
Effects	Filter, Time, Frequency-based effects
Performance	Foot, Hand, Breath, Band, Sing
Control	Accent, Volume, Sound source
Notation	Note, Rest, Key, Clef, Staff, Bind, Tie, Slur, Text, Dotted Note, Sharps and Flats
Lyrics	Lyric Element, Bars and Lyric Text, Stretch and Squash
MIDI	MIDI Controllers

5. JScoreML

JScore는 음악을 재생하면서 온라인 상으로 악보를 표현하면서 편집까지 지원하는 Applet 프로그램이다. 따라서 특정 음악을 재생하면서 MIDI용 파일로 해당 음악을 생성해 낼 수도 있다[12]. 표 5는 JScoreML에서 정의된 음표(Note)에 대한 DTD를 표현하였다. Note 요소(Element)에 대한 속성으로 여러 가지 음악적 구성요소들이 사용되고 있다. 그러나 JScoreML은 자바 애플릿 프로그램이라는 특성으로 반드시 자바 API들이 필요하며, JAXP XML 파싱 API만이 사용되어져서 범용성(Generalization)이 떨어진다.

표 5. JScoreML DTD
Table. 5 JScoreML DTD

```
<!ELEMENT note EMPTY>
<!ATTLIST note id   ID #REQUIRED
  value    (c|d|e|f|g|a|b)          #REQUIRED
  octave   (0|1|2|3|4|5|6|7|8)  "4"
  acciaccatura (none|flat|sharp) #IMPLIED
  duration (1|2|4|8|16|32|64)  "4"
  point    (single)    #IMPLIED
  legato   IDREF       #IMPLIED
  dynamic
    (ppp|pp|p|mp|mf|f|ff|fff) #IMPLIED
  crescendo IDREF  #IMPLIED
  diminuendo  IDREF  #IMPLIED>
```

6. XScore

악보의 내용을 저장하거나 네트워크 상에서 전송하는 데 일반적이고 표준적인 접근법을 제공하는 것이 주 목적인 XScore(eXtensible Score Language)는 XML의 응용이면서 악보의 구조와 내용을 표현하기 위한 언어[13]이다. 특징은 기타 연주를 위한 타브(Tablature) 악보 표현까지 지원하고 있으며 표 6은 XScore의 DTD 구조 중에서 음표에 관련된 사항을 보여주고 있다. 음표당 표현되는 속성의 수가 다수인 것을 알 수 있고, 표 7은 그림 4를 SXcore로 표현한 XML 인스턴스이다. 현재 성가(Chant)와 퍼커션(Percussion)을 위한 악보 표현을 연구하고 있다.

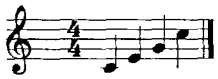

그림 4. 예제 악보
Fig 4. Example Sheet

표 6. XScore DTD
Table. 6 XScore DTD

```
<!ENTITY
    % attr-notation  "id ID          #IMPLIED
    text             CDATA          #IMPLIED
    d  (1|2|4|8|16|32|64|128)  #IMPLIED
    dots             (1|2|3)        #IMPLIED
    accent           .생략.         #IMPLIED
    dynamics         .생략.         #IMPLIED
    ..... 생략 .....
    tie-slur         IDREF          #IMPLIED
    bow              (up|down)      #IMPLIED
    ped-text         CDATA          #IMPLIED">
<!ELEMENT    c                      (note*)>
```

표 7. XScore 표현
Table. 7 Expression of XScore

```
<xscore>
  <title>"C" Arpeggio</title>
  <composer>R. J. Grigaitis</composer>
  <staff><bar>
    <clef clef="treble"/>
    <key key="C"/>
    <time beats="4" note="4"/>
    <voice>
      <n p="1" d="4"/>
      <n p="3" d="4"/>
      <n p="5" d="4"/>
      <n p="1o1" d="4"/>
    </voice></bar></staff></xscore>
```

7. MusicXML

Recordare LLC사에서 2001. 11월에 베타 버전으로 발표한 악보표현기법이다. GUI(Graphic User Interface)환경에서 편리하게 악보를 편집, 수정할 수 있는 기능을 가지고 있으며, 특히 17세기 이전의 음악을 표현하기 위한 구조이다[14]. MusicXML의 주요 DTD는 Partwise.dtd와 Timewise.dtd를 제공하고 있으며, 이에 대한 XSLT 스타일시트 또한 제공되고 있다. 또한 악장(Movement)의 진행을 위한 "opus.dtd"가 있으며, XLinks 구조를 사용하여 상세한 정보는 참조 형태로 사용하고 있으며, MIDI용 표현까지 제공하고 있다. 또한 다양한 음악 표현 기법(Notation)간 분석과 검색 그리고 연주용 응용프로그램 간 상호 교환 형식으로 사용할 수 있다. 그림 5는 MusicXML의 기능을 설명하며, 표 8은 MusicXML에서 음표를 표현하기 위해 정의된 DTD 내용이며 또한 표 9는 그림 6을 DTD 구조에 의해 표현한 것이다.

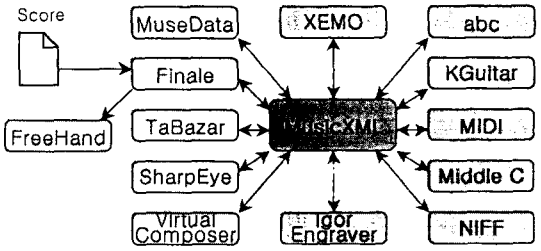

그림 5 MusicXML 기능
Fig. 5 Functions of MusicXML

표 8. MusicXML DTD
Table. 8 MusicXML DTD

```
<!ENTITY
    % full-note "(chord?, (pitch | rest))">
<!ENTITY
    % voice-track "(footnote?, level?, track?)">
<!ELEMENT
    note ((((cue | grace), %full-note;) |
    (%full-note;, duration, tie?)), %voice-track;,
    type?,dot*,accidental?, time-modification?,
    stem?, staff?, beam*, notations*, text*)>
```

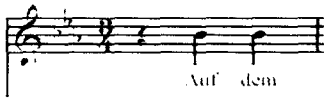

그림 6. 예제 악보
Fig. 6 Example Sheet

표 9. MusicXML 표현
Table. 9 Expression of MusicXML

```
<note>    <pitch><step>B</step>
              <alter>-1</alter>
              <octave>4</octave>
          </pitch>
 <duration>24</duration>
 <voice>1</voice>
 <type>quarter</type>
 <stem>down</stem>
 <lyric number="1"><syllabic>single</syllabic>
     <text>Auf</text> </lyric>
</note>
<note> .. 생략 .. </note>
```

8. 기존 연구의 악보 표현 메카니즘

위에서 살펴본 기존 연구들과 더불어 다양한 목적들을 위해 MusiqueXML, ChordML, EMNML, XMusic, MDL, VMML 등과 같은 악보 표현법들이 소개되고 있다. 음악 정보를 표현하기 위해 수 많은 DTD 또는 스키마가 정의되고 있는 것은 그만큼 표준화된 표현법이 부재이며, 활용 목적에 따라 표현법 또한 다양하게 된 것이다. 또한 음악 정보를 모든 음악적 요소들에 초점을 두고 표현하고 있어서 모든 정보는 각각 독립되어 있다는 것이다. 그래서, XML 문서 인스턴스상에 태그의 수와 또한 물리적인 데이터베이스내로 저장하는 단계에서는 필드의 수(엘리먼트)가 유동적으로 변하게 되고, 특정 음표만을 검색해야 하는 경우에는 검색 질의 단계가 자연히 깊어지고, 반복 검색을 해야 하는 상황이 발생하게 된다.

III. 스키마 모델링

모델링을 위한 전제 조건은 다음과 같다.

- ▶ 대보표(Great Staff) 기준.
- ▶ 소절 단위 정의
- ▶ 서양음악(Western Style Music) 기준
- ▶ 관련 음악 정보는 그룹으로 처리
- ▶ 세부 음악 정보는 기존 연구 참조
- ▶ 스키마 모델은 베니션 블라인드 모델 사용

관련 음악 정보라는 것은 음표를 표현해야 한다면 한 소절 안의 음표들을 그룹으로 처리한다는 것이며, 또 다른 음악정보들 또한 이와 같은 방법으로 처리한다는 것을 말한다. 그리고 세부 음악 정보들을 표현하기에는 본 논문의 향후 연구 내용들이 될 것이며, 여기서는 기본적인 중요 요소들에 대한 모델링을 제안 한다.

1. 소절(Measure) 단위

악보는 음의 높낮이를 표현하기 위해서 5개의 선으로 이루어진 오선 상에 연주되는 악곡의 악기편성이나 연주형태 등을 기보한 오선 보표이며 편성에 따라 수십 단에 이른다. 따라서 악보는 크게 총보(Score), 대보표, 파트 보(Part Sheet), 리드시트(Read Sheet), 마스터 리듬시트(Master Rhythm Sheet), 타브 시트(Tablature) 등으로 구분할 수 있다. 그리고 악보의 기본 구조는 동기(Motive), 악구(Phrase, 작은악절), 악절(Period, 큰악절), 주제(Theme)로 구성된다. 따라서 본 논문은 악보의 기본 구조가 되는 동기의 절반인 소절을 하나의 단위로 취급하여 모든 음악적 요소들의 정보를 표현하게 된다. 즉, 기존의 표현 방식들에서는 각각의 음악적 요소들에 초점을 두고 있지만 제안하는 방식에서는 모든 표현이 소절단위로 처리된다. 다음 사항은 악보에 표현되는 일반적인 구성 요소들을 제안하는 방식에서 사용하기 위해 크게 두 부분으로 나누어 표현했다.

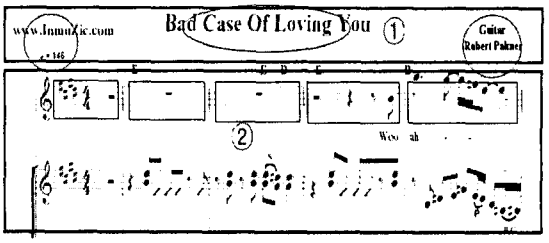

그림 7. 예제 악보
Fig. 7 Example Sheet

그림 7상에서 ①과 ②에는 각각 다음과 같은 음악적 구성 요소들이 나타날 수 있으며, 특히 ②에서 보면 5개의 소절이 하나의 악구를 구성하고 있으며 아래의 구분란의 구성요소 중 기타구성요소들은 일반적인 표현 사항 이외에 사용자가 특별히 추가하고 싶은 사항 등을 말한다. 그리고 이것은 모델링 시에 이를 표현할 요소를 옵션으로 처리할 수 있으므로 각각의 경우를 나타내지 않았다.

표 10. 구성 요소 분류
Table. 10 Classification of Component

구 분	구성 요소
서두부분 (Head)	곡제목, 부제목 작곡자, 작사자, 편곡자, 가수, 나타냄말(곡상), Key, 기타구성요소들
몸체부분 (Body)	보표, 음자리표, 조표, 임시표, 박자, 빠르기표, 셈여림표, 나타냄말, 주법, 음표, 쉼표, 잇단음표, 줄임표, 꾸밈음, 기타 구성요소들

2. 표현될 음악적 구성 요소

일반적인 악보를 XML DTD나 스키마로 정의할 때 필요한 기본 구조는 표 10에서 살펴본 바와 같이 곡 제목, 작곡자, 작사자, 가수 등과 같이 악보의 서두 부분에 표현되어지는 정보와 보표 상에 표현되는 여러 가지 음악적 구성요소들이 된다. 본 논문에서는 대보표(높은 음자리표와 낮은 음자리표를 함께 묶어서 사용하는 보표, 원곡과 피아노 파트로 구성)를 기준으로 하며, 그림 7과 표 10에 구분된 내용에 따라 머리부분(Head)은 악보의 가운데 부분(Head_Center)과, 왼쪽부분(Head_Left), 그리고 오른쪽 부분(Head_Right)으로 나누도록 하고 몸체부분(Body)부분은 악보 전체를 구성하는 가장 작은 단위인 소절 단위로 구성이 되며 여기에 일반적인 악보상의 음악적 구성요소들이 표현되어지도록 한다. 또한 대보표가 기준이므로 소절들은 원곡 파트와 피아노 파트로 구성되도록 한다. 음악적 구성요소들을 정리하면 다음과 같다.

3. Head 부분의 구성 요소

1) Head_Center

이곳은 악보의 제목과 부제목이 표현될 수 있다. 따라서 이를 표현하기 위하여 스키마 선언 시 <Head_Center>를 사용하고 <Title>과 <Sub_Title>을 두도록 한다.

2) Head_Left

이곳에는 일반적으로 곡에 대한 곡상(Expression Signature), 또는 빠르기표(Tempo Signature, Metronome) 등이 속하며, 각각의 구성 요소들에 대하여 DTD상에서는 엔티티 방식, 그리고 스키마에서는 ComplexType으로 선언하여 사용하도록 한다.

3) Head_Right

이곳에는 악보에 대한 작사자, 작곡자, 가수, 편곡자 등의 정보를 표현할 수 있으며 각각의 구성요소들을 별도로 정의하기 보다는 하나의 ComplexType으로 선언하여 사용하도록 한다.

4. Body 부분의 구성 요소

Body에서 표현될 구성 요소의 내용은 크게 두 부분이다. 원곡을 표현하는 부분(높은 음자리표)과, 원곡을 피아노곡으로 편집한 부분(낮은 음자리표)이며 각각의 부분에는 악보를 작성하는 데 쓰이는 여러 가지 음악적 구성 요소 및 기호(악전정보)들이 불규칙하게 나타나게 된다. 아래에 본 논문에서 제안한 소절 단위의 리모델링을 위해 필요한 음악적 구성요소들을 기술한다.

1) 소절 순서

소절은 동기를 구성하며, 또한 악구와 악절을 구성하는 기본 단위이다. 즉, 소절의 연속된 집합이 하나의 악보를 구성하게 된다. 따라서 소절의 순서가 보표(Staff)별로 필요하고 각각의 소절별로 악보상에 표현된 음악적 구성 요소들이 그룹화 된다. <Measure> 요소에 "part" 속성을 이용하여 보표를 구분하고 반복되는 사항을 위해 "repeat" 속성을 사용하도록 한다.

2) 보표

소절 정보가 어느쪽 보표 정보인지를 나타내기 위해서는 보표를 구분하는 것이 필요하다. 보표는 대보표이므로 피아노 파트(높은 음자리표)와 연주 파트(낮은 음자리표)를 사용한다.

3) 음자리표(Clef)

음자리표는 높은 음자리표(G Clef)와 가온음자리표(C Clef), 그리고 낮은 음자리표(F Clef)가 기본인데, 본 논문에서는 높은 음자리표와 낮은 음자리표를 보표의 파트를 기준으로 정하도록 한다.

4) 조표(Key)와 임시표(Accidentals)

조표는 곡의 처음에 표시되어 이 곡이 무슨 조에 속하는 곡인지를 명시하는데, 악전 기호에 대해 기호화된 문자를 사용한다. 즉, 다음과 같다.

▶ C,D,E,F,G,A,B(♭ , #)

그리고 임시표는 악보의 중간에 임시로 어떤 음을 변화시킬 때 사용된다. 즉, 조표와 임시표는 모든 소절에 포함될 가능성이 있으므로 이를 표현하는 스키마 구조가 정의 되도록 한다.

5) 박자표 (Time Signature)

악보의 첫머리, 즉 첫 번째 소절에 일반적으로 숫자로 표현된다. 이는 박자의 수와 곡의 단위 음표로 구성되며 형태는 3/4, 4/4 등이다. 박자의 수와 단위 음표를 서로 분리하지 않고 하나의 정보로 표현하도록 한다.

6) 빠르기표 (Tempo Signature)

곡의 빠르기를 표시하는 것이며 일반적으로 Metronome의 앞 글자나, 곡의 빠르기를 나타내는 말을 곡의 첫머리에 표현하기도 하고, 곡의 중간에 표현되기도 한다. 두 가지 경우를 허용하도록 한다. 즉, Head_Left에 나타날 수도 있다.

7) 셈여림표 (Dynamics Signature)

곡의 강약 기호, 또는 말을 나타내는 것으로서, 곡의 중간, 또는 시작 부분에 올 수 있다.

8) 나타냄말 (Expression Signature)

곡상을 표현하는 말로서, 곡의 중간 또는 시작 부분에 올 수 있다.

9) 주법 (Play Pattern)

연주법을 나타내는 말로서, 레카토(Legato : Slur, Tie), 스타카토(Staccato), 페르마타(Fermata) 등 다양하다. 곡의 중간 또는 시작 부분에 올 수 있다.

10) 음표와 쉼표(Rest)

음표는 악보 상에 가장 많이 나타나는 기호이며, 다양한 추가 의미들이 포함된다. 일반적인 악보 상에 음표는 단순히 음표 하나일 수 있고, 또는 추가 의미들이 첨가되는 경우(C#, C♭) 등이 있고, 또한 음표는 음표의 길이(♪: 8분음표)가 있어서 기본 음표, 점음표, 겹점 음표, 잇단 음표(민음표, 점음표) 등이 사용된다. 따라서, 모든 음표에 대하여 음표에 해당하는 숫자를 사용하면서 구분자를 같이 사용하여 표현한다. 또한 점음표와 겹점 음표들은 기본 음표에 절반이거나 혹은 하나 반이기 때문에 괄호를 사용하여 표 11과 같은 방식으로 표현한다. 그리고 음계(Scale)는 "CEDFGAB"를 사용하고, 반음인 경우와 반올림인 경우에는 "-,+"기호를 사용한다. 또한 옥타브(Octave)는 악기마다 서로 다르지만 피아노를 기본으로 했을 때 피아노의 구성은 전체 건반이 8옥타브로 구성되어 있으므로 순서대로 "1,2,3,4,5,6,7,8, 9,A"로 표현한다. 그리고 피아노곡에 대한 표현 범위가 원곡의 표현 범위를 포함하므로 원곡에 대한 표현은 피아노곡의 표현을 그대로 사용하도록 한다. 즉, 원곡에 대하여 피아노곡의 멜로디는 가온다 음을 기준으로 왼손으로 2옥타브 아래와 오른손으로 2옥타브 위를 일반적으로 사용하여 표현되기 때문이다. 또한 음표의 기둥(Stem)이 윗쪽인지, 아래쪽인지를 구분하기 위하여 기존의 연구들에서는 <stem> 태그 등을 사용하고 해당 값을 표현하고 있지만 본 연구에서는 대보표를 사용하고 있기 때문에 보표 상에서 최대 옥타브와 최저 옥타브를 기준으로 기둥이 위로 향하는지, 아래로 향하는지를 결정하도록 한다. 음표의 길이는 표 11과 같다. 정리하면 하나의 음표가 표현되는 순서는 악보에 표현되는 순서를 지키도록 하는데, 피아노의 기본 "도"의 4분음표는 "(4C4)"로 표현(옥타브 음계 음의 길이)되며 4분 점음표는 "(4C(4:8))"가 된다. 쉼표는 표 12와 같은 종류가 있으며, 음표 길이 표현과 동일한 방법으로 표현하고 단지 요소명으로 구분되도록 한다.

표 11. 음의 길의 표현
Table. 11 Expression of Tone Length

음표종류	음표	참고	음표 표현
기본음표	온음표		1
	2분음표		2
	4분음표		4
	8분음표		8
	16분음표		16
	32분음표		32

음표종류		음표	참고	음표 표현
점 음 표		점온음표	(1:2)	3
		점2분음표	(2:4)	6
		점4분음표	(4:8)	12
		점8분음표	(8:16)	24
		점16분음표	(16:32)	48
겹점 음표		겹점온음표	(1:2:4)	7
		겹점2분음표	(2:4:8)	14
		겹점4분음표	(4:8:16)	28
잇단 음표	민음표	셋잇단음표	(4 (3)),(8 (3)), (16 (3))	
		다섯잇단음표	(8 (5)),(16 (5)), (32 (5))	
		여섯잇단음표	(8 (6)),(16 (6)), (32 (6))	
		일곱잇단음표	(8 (7)),(16 (7)), (32 (7))	
	점음표	두잇단음표	(2 (2)),(4 (2)),(8 (2))	
		네잇단음표	(2 (4)),(4 (4)),(8 (4))	
		다섯잇단음표	(2 (5)),(4 (5)),(8 (5))	
		일곱잇단음표	(4 (7)),(8 (7)),(16 (7))	
		여덟잇단음표	(4 (8)),(8 (8)),(16 (8))	

표 12. 쉼표의 표현
Table. 12 Expression of Rest

쉼표 종류	쉼표 표현
온쉼표	1
2분쉼표	2
4분쉼표	4
8분쉼표	8
16분쉼표	16
32분쉼표	32

11) 옥타브(Octave)

가온 다를 4옥타브라 한다. 즉, 가온 다 옥타브의 윗 옥타브는 5 옥타브가 되고 밑의 옥타브는 3옥타브로 정한다.

12) 슬러(Slur)

<slur> 요소를 사용하며 음표의 진행 순서상의 구분을 위해 ":" 기호와 "start", "end"값으로 슬러를 표현하도록 한다.

13) 보(Stem)

<stem> 요소를 사용하며 음표의 진행 순서상의 구분을 위해 ":" 기호와 "start", "end"값으로 보를 표현하도록 한다.

14) 화음(Chord)

3화음과 7화음, 또는 화음 밖의 음 등이 존재하며 악보상에 표현된 화음을 표현하기 위해 <note> 요소의 텍스트 값으로 "H"를 사용하고, 음표의 순서는 아래에서 위쪽 순서로 정하도록 한다.

15) 가사(Lyric)

가사는 일반적으로 1옥타브 반 정도를 사용하는데 이미 가사에 대한 음표 표현은 10)절에서 설명하였으므로, 소절 당 가사만을 순서대로 표현하기만 하면 된다. 특정 음표에 해당되는 가사가 없는 경우에는 "-"를 추가하도록 한다.

16) 구분자(:)

이는 음악 정보들이 그룹으로 묶여지게 되면 구분자는 음악 정보들 간의 구분을 위해 사용될 기호가 된다. 예를 들어서, 한 소절 안에 음표가 4개 들어 있다면 "(CFGE)"가 "(C:F:G:E)"로 표현하기 위해 각각을 구분하기 위한 목적으로 사용한다.

5. 스키마 구성 모델

XML 스키마와 네임스페이스(Namespace)는 서로 밀접한 관계를 가지고 있다. 중복된 정의를 서로 구분하기 위한 개념인 네임스페이스를 고려한다면, 스키마의 특정 요소들을 정의할 때 3가지 모델 설계 기법에 기초하게 되는데 그 모델은 러시아 인형 설계법(Russian Doll Design), 살라미 조각 설계법(Salami Slice Design), 베니션 블라인드 모델(Venetian Blind Model)이다. 스키마 구성 모델은 재사용의 효율성을 증대하기 위하여 본 연구에서는 베니션 블라인드 모델을 기본 모델로 채택한다. 그리고 표 13에서 보여주듯이 음악적 구성 요소들은 SimpleType 형식과 ComplexType 형식들로 구분되어 정의된다. 즉, 다른 요소를 포함할 수 있는 음악적 구성 요소들은 ComplexType으로 정의하며, 그렇지 못하는 요소들은 SimplexType으로 정의된다.

표 13. 데이터형
Table. 13 Data Types

형 식	음악적 구성 요소
SimplexType	가사, 쉼표, 음표, 슬러, 보, 구분자
ComplexType	Head 부분 구성 요소, 소절 정보

즉, 베니션 블라인드 모델은 단순한 요소 선언이 아닌 ComplexTypes의 사용을 권한다. 이 접근법으로 요소를 정의하면 정의된 요소들은 재사용을 할 수 있게 해준다. 즉, 클래스를 정의하고 객체를 생성할 때 그 클래스를 사용하는 것에 비유될 수 있다. 따라서 본 논문에서 제안하는 방식에서 Head 구성 요소들 이외에 소절(Measure)을 ComplexType으로 정의하는 것은 XML의 네임스페이스(Namespace)와 관련하여 이 소절을 재사용의 단위로 사용할 수 있도록 하기 위함이다.

IV. 모델링 결과

1. 제안 모델

3절에서 기술된 내용을 기반으로 본 논문에서 제안하는 악보 표현의 XML 스키마 모델은 표 14와 같다.

표 14 제안 스키마
Table. 14 Proposed Schema

```
<?xml version="1.0"?>
<xs:schema 서두 부분 생략 ......>

<xs:element name="Sheet" type="SheetType">
<xs:element name="Head" type="HeadType">
<xs:element name="Body" type="BodyType">
<xs:element name="HeadLeft" type="HeadLeftType">
<xs:element name="HeadCenter" type="HeadCenterType">
<xs:element name="HeadRight" type="HeadRightType">

<xs:complexType name="SheetType">
   <xs:sequence>
      <xs:element name="Head" type="HeadType">
      <xs:element name="Body" type="BodyType">
   </xs:sequence>
</xs:complexType>

<xs:complexType name="HeadType">
  <xs:sequence>
  <xs:element name="HeadLeft" type="HeadLeftType">
  <xs:element                    name="HeadCenter"
type="HeadCenterType">
  <xs:element name="HeadRight" type="HeadRightType">
  </xs:sequence>
</xs:complexType>

<xs:complexType name="HeadLeftType">
   <xs:choice>
      <xs:element name="Expression" type="xs:string">
      <xs:element name="Key" type="xs:string">
   </xs:choice>
</xs:complexType>

HeadCenter, HeadRight 생략....

<xs:complexType name="BodyType">
   <xs:element name="Measure" type="MeasureType">
   <xs:attribute name="Order" type="xs:string"/>
   <xs:attribute name="Repeat" type="xs:boolean"/>
   <xs:attribute name="Part" type="xs:string"/>
</xs:complexType>

<xs:complexType name="MeasureType">
   <xs:choice>
      <xs:element name="Note" type="xs:string" />
      <xs:element name="Slur" type="xs:string" />
      <xs:element name="Beam" type="xs:string" />
      <xs:element name="Lyric" type="xs:string" />
   </xs:choice>
</xs:schema>
.. 이하 생략..
```

2. 결과

제안한 모델링 방식의 결과를 위하여 "조수미"가 부른 "Champions(월드컵 노래)"의 그림 8의 악보를 기존 연구 방식(MusicXML)과 제안한 연구 방식으로 구분하여 XML 문서 인스턴스로 표현하였으며, MusicXML과 표현 방식을 비교하는 이유는 관련 연구 부분의 그림 5에서 살펴본 것처럼 MusicXML은 현재 여러 가지 응용 애플리케이션이나 표현 방식들 간에 있어서 상호 교환 형식으로 사용되고 있기 때문이다. 따라서 MusicXML과 제안된 모델링 방식의 비교 결과를 근거로 MusicXML의 표현 방식을 새롭게 구성할 수 있는 계기가 되기 때문이다.

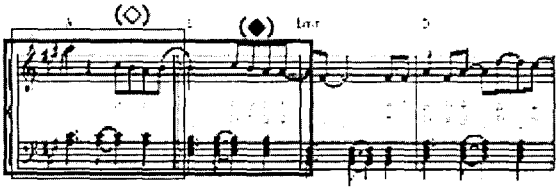

그림 8 예제 악보
Fig. 8 Example Sheet

1) 기존 연구 방법을 이용한 표현(MusicXML)

그림 8의 (◇) 부분의 "너-와"까지 표현한 내용이 표 15이다. 즉, 첫째 소절의 절반을 표현한 내용이 된다. 표 15의 내용을 살펴보면 모든 정보가 음표단

위로 표현되는 것을 볼 수 있다. 또한 표현되는 정보들도 서로 독립적인 형태를 유지하고 있다.

표 15 MusicXML 표현
Table. 15 Expression of MusicXML

```
<measure number="9">
<note><pitch> <step>A</step>
                <alter>1</alter>
                <octave>5</octave>
      </pitch>
      <duration>24</duration>
      <voice>1</voice>
      <type>quarter</type>
      <stem>down</stem>
</note>
<note>
      <rest/>
      <duration>24</duration>
      <voice>1</voice>
      <type>quarter</type>
</note>
<note><pitch> <step>C</step>
              <alter>1</alter>
              <octave>5</octave>
         </pitch>
        <duration>12</duration>
        <voice>1</voice>
        <type>eighth</type>
        <stem>down</stem>
        <beam number="1">begin</beam>
         <lyric number="1">
                <syllabic>begin</syllabic>
                <text>너</text>
         </lyric>
</note>
<note><pitch> <step>B</step>
              <octave>5</octave>
         </pitch>
        <duration>12</duration>
        <voice>1</voice>
        <type>eighth</type>
        <stem>down</stem>
          <lyric number="1">
                <syllabic>middle</syllabic>
        <text> - </text>
          </lyric>
</note>
<note><pitch> <step>A</step>
              <octave>5</octave>
         </pitch>
        <duration>12</duration>
        <voice>1</voice>
        <type>eighth</type>
        <stem>down</stem>
         <lyric number="1">
                <syllabic>middle</syllabic>
              <text>와</text>
         </lyric>
</note>
<note> .. 이하 생략 ..  </note>
</measure>
... 이하 생략
```

2) 제안한 모델을 이용한 표현

그림 8의 (◆) 부분의 "너-와나-지금여기"까지 표현한 내용이 표 16의 <Measure order="9" part="piano"'> 이후의 내용이다. 즉, 두 번째 소절까지 표현한 것이다. 서로 관련되는 정보는 통합하여 하나의 소절단위로 표현한 결과가 된다.

표 16. 제안된 방식의 표현
Table. 16 Expression of Proposed Model

```
XML 서두 부분 생략
<Sheet genre="public">
  <Head>
    <HeadLeft> ..... </HeadLeft>
    <HeadCenter> ..... </HeadCenter>
    <HeadRight> ..... </HeadRight>
  </Head>

  <Body>
   <Staff part="piano">
     key, time, clef 기술
   </Staff>
   <Staff part="performance">
     key, time, clef 기술
   </Staff>

   <Measure order="1" part="piano">
    <note>(R1)</note>
   </Measure>

   <Measure number="1" part="performance">
    <note>3(F+12:F+8:F+4:F+4)(</note>
    <slur>(:start:end:)
   </Measure>

   <Measure order="2" part="piano">
    <note>(4A+2)(R2)(4A+2)</note>
   </Measure>
   <Measure number="2" part="performance">
    <note>3(C+12:C+8:C+4:C+4)(</note>
    <slur>(:start:end:)
   </Measure>

   ..... 이하 생략

   <Measure order="9" part="piano">
    <note>(4A+4)(R(2))(4(C:B:A+:B)8)</note>
    <slur>(;::::start)</slur>
    <beam>(::start::end)</beam>
    <lyric><::너-와나</lyric>
   </Measure>

   <Measure order="9" part="performance">
    <note>
     H(2F+4:2A+4):H(2F+8:2A+8):H(2F+8:2A+8):
     H(2F+4:2A+4)</note>
    <slur>(:start:end:)</slur>
   </Measure>
```

```
<Measure order="10" part="piano">
 <note>(4B2)(4(C:B:A+:A+)8)</note>
 <slur>(end::::start)</slur>
 <beam>(:start::end)</beam>
 <lyric><-지금여기</lyric>
</Measure>
<Measure order="10" part="performance">
 <note>
    H(2C+12:2E+12:2G+12):H(2C+8:2E+8:2G+8):
    H(2C+8:2E+8:2G+8):H(2C+4:2E+4:2G+4)
 </note>
 <slur>:start:end:</slur>
</Measure>
..... 이하 생략
</Body>
</Sheet>
```

3) 모델링 결과 분석

그림 8에 대한 MusicXML 표현인 표 15와 제안한 모델로 표현한 표 16의 XML 인스턴스 내용을 살펴보면 제안된 모델은 그림 8의 음악 구성 요소들에 대하여 XML로 인스턴스가 가능하면서 문서 정의를 XML 스키마로 확장했으며, 특히 베니션 블라인드 모델 정의에 의한 스키마 정의로 스키마 재사용성 증대를 가져올 수 있게 되었다. 또한 적용 분야에 따라 다를 수 있지만 상용 음악 정보 표현들이 대보표 기준인 점을 감안할 때 제안된 방식 표현이 선호될 수 있으며, 표현 핵심에서 보면, 기존 연구들의 공통점인 단일화(개별화) 특성의 표현으로 인한 방대한 XML 문서 인스턴스의 분량(평균 15소절 표현 시 4,000라인)을 그룹화된 표현으로 평균 1/2 정도 이상 줄일 수 있고, 또한 XPATH를 이용한 음악 정보 검색 시 XPATH 검색식 단순화의 기대와 애플리케이션의 도움 없이 사람이 직접 표현할 수 있다는 이점을 가질 수 있을 것으로 본다. 비교 결과 분석은 표 17에 정리하였으며, 제안한 방식에서 음표 분석 흐름도와 <note> 태그의 String 분석 알고리즘은 그림 9와 그림 10에 제시하였다.

표 17. 결과 분석
Table. 17 Result Analysis

분 야	MusicXML	제안된 방식
XML 표현	가 능	가 능
구 조	DTD	스키마
재 사 용	복 잡	단 순
분 야	총 보	대보표
표현 핵심	단일화	그룹화
검색식(XPATH)	복 잡	단 순
표현 능력	기계에 의존	사 람
인스턴스 분량	방 대	평균 1/2

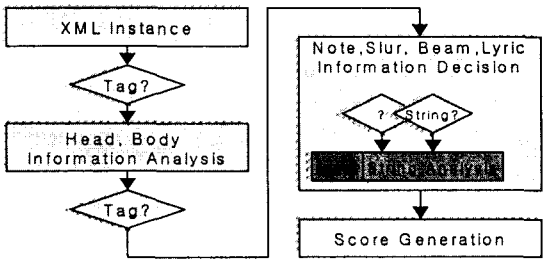

그림 9. 분석 흐름도
Fig. 9 Flow Diagram of Analysis

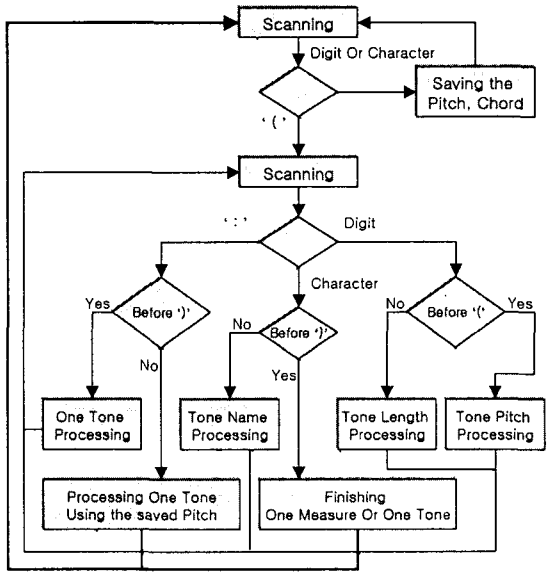

그림 10. Note 태그 내 String 분석 알고리즘
Fig. 10 Algorithm of String within Note

V. 결 론

본 연구에서는 XML이 가지고 있는 간단성, 확장성, 재사용성의 장점을 이용하여 기존의 음악 정보를 표현하기 위한 XML DTD, 스키마 모델을 소절 단위로 그룹화하여 표현하여 기존 모델을 새롭게 모델링 하였다. 음악의 복잡성으로 인해 아주 다양한 표현 방식들이 존재하고, 표준화 되지 않은 문제점들과, 적용 분야의 다양성으로 인해 그 표현 모델은 모델마다 상이하다. 본 논문은 이러한 상황에서 대보표를 기준으로 관련된 음악정보들을 통합화하여 표현하는 모델을 제안하였다.

본 논문의 향후 연구과제는 세밀한 음악정보 구성요소들에 대한 추가 표현 연구가 필요하며, 표현된 XML 인스턴스 내용을 그래픽 악보로 표현하는 응용프로그램 개발과 총보표를 위한 스키마의 확장 연구가 필요하다. 또한 기존의 XML을 이용한 음악 표현 모델들과의 표준화가 필요하다.

참고문헌

[1] 김태완, 배미숙, 황부현, "효율적인 선율 검색을 위한 XML 문서의 저장소 설계", 한국정보처리학회 할술지, Vol. 8, No. 1, 2001.

[2] http://www.w3.org/TR/REC-xml-19980210

[3] Dongwook Shin, Hyuncheol Jang, and HongLan Jin "Bus:An Effective Indexing and Retrieval Schemaa in Structured Documents", ACM. pp 235-243, 1998.

[4] Francois. "Generalized SGML repositories:Requirements and Modeling", Computer Standards & Interfaces, 1996.

[5] Tuong Dao, Ron Sacks-Davis, James A.Thom. "An indexing scheme for structured documents and its implementation", Proceedings of the 4th International Conference on DATABASE Systems for Advanced Applications, Melbourne, Australia. pp. 125-135, 1997.

[6] 맹성형, 주종철 "문서 구조화와 정보 검색", 정보과학회지, 제 16권, 제8호, 1998.

[7] http://www.cned.or.kr/jnjmuse/compose1_-low.htm

[8] http://www.techno.com/smdl.htm

[9] http://www.oasis-open.org/cover/-mnmlv200.html

[10] http://www.shigeta.net/mhtml/index.htm

[11] http://www.musicmarkup.info/

[12] http://nide.snow.utoronto.ca/music/-index.html

[13] http://fn2.freenet.edmonton.ab.ca/~rgrigait/-xscore/

[14] http://www.musicxml.org/xml.html

저자소개

김정희(Jeong-Hee Kim)

1994년 제주대학교 정보공학과 졸업(학사)

1997년 제주대학교 대학원 정보공학과 졸업(석사)

2002년 제주대학교 대학원 정보공학과 박사과정

1998년~현재 제주산업정보대학 컴퓨터정보계열 겸임교수

※관심분야: XML, 데이터베이스, Internet Application. 프로그래밍 언어론, GIS 등

곽호영(Ho-Young Kwak)

1983년 홍익대학교 전자계산학과 졸업(학사)

1985년 홍익대학교 대학원 전자계산학과 졸업(석사)

1991년 홍익대학교 대학원 전자계산학과 졸업(박사)

1990년~현재 제주대학교 통신컴퓨터공학부 교수

※관심분야: 객체지향 프로그래밍, 프로그래밍 언어론, GIS 등