XML 기반 리듬 편집 및 재생 시스템 개발

손 원 성† · 이 용 규††

요 약

지금까지 음악 정보를 컴퓨터상에서 표현하기 위한 많은 연구가 진행되었다. 그러나 기존의 방법들은 독자적인 표현 방식을 사용하므로 서로 호환되지 않는 문제점을 갖고 있다. 따라서 본 논문에서는 이를 해결하기 위하여 음악정보, 특히 리듬 정보를 웹 표준 언어인 XML을 이용하여 표현하고 이를 편집 및 재생하기 위한 새로운 방법을 제시하고 편집 및 재생 시스템을 개발한다. 본 논문에서 정의한 리듬 정보의 표현은 기존의 매우 복잡한 음악 정보 표현 방법과는 달리 RDML이라는 언어를 통하여 리듬 정보를 간결하게 표현하는 동시에 리듬의 비트, 악기의 종류, 출력 정보와 같은 리듬 정보를 정확하게 표현할 수 있다. 또한 RDML 리듬 정보는 SMIL로 변환하여 재생이 가능하다. 우리의 방법은 웹 표준에 근거하였으므로 웹에서의 음악정보의 공유가 가능하다.

Development of a Rhythm Editing and Playing System Based on XML

Won Sung Sohn† · Yong Kyu Lee††

ABSTRACT

Much research has been performed to represent music information on computers. However, one problem of the previous approaches is that music information cannot be shared with other music systems since they have used their own representation schemes. In order to resolve this problem, we have developed a new scheme for representing, editing, and playing rhythms among music information based on XML which is a web standard language. We present a RDML (Rhythm Description Markup Language) to exactly describe rhythm information such as beats, musical instruments, and performance information, which is much simpler than previous representation schemes. Moreover, the RDML representation can be played by converting it into SMIL representation. Because our approach is based on the web standard, the music information can be shared on the web.

1. 서 론

음악 정보를 디지털 형태로 표현하여 컴퓨터에 적용시키기 위한 시도는 1949년 Bronson[9]의 연구로부터 시작되었다. 그 이후 음악 정보에 대한 컴퓨터 응용은 작곡[10], 편집[11], 검색[20] 등 다양한 분야로 확대되었으나 음악 정보를 컴퓨터 상에서 완벽하게 처리할 수 있는 연구 성과를 거두지는 못하였다. 그 이유는 음악 정보를 전산화하는 문제가 예상보다는 매우 복잡하기 때문이다. 예를 들어 음의 길이, 높이, 세기, 빠르기 등을 결정하는 선율은 1,200개 정도의 음악기호로 구성되어 있으며 이러한 기호들이 발생하는 경우가 다양하기 때문에 모든 경우를 처리하기 위한 완벽한 방법을 구현하는 것은 매우 어려운 작업이라 할 수 있다[12, 15]. 그러나 현재까지도 음악 정보를 처리하기 위한 여러 가지 연구가 시도되고 있으며 음악 정보 처리를 위한 다양한 상용 제품도 개발되고 있다.

† 준 회 원 : (주)사이버테크 연구원
†† 종신회원 : 동국대학교 컴퓨터 · 멀티미디어공학과 교수
논문접수 : 1999년 12월 23일, 심사완료 : 2000년 5월 3일

음악 정보를 디지털 형태로 표현하기 위한 기존 연구 방법들은 코드화 방법을 이용한 것들이 대부분이다. 이러한 방법들 가운데 현재 표준 규격으로 채택된 것으로는, ISO에서 SGML과 HyTime을 이용하여 정의한 SMDL(Standard Music Description Language)[1], 음악 관련 소프트웨어 제작자들과 연구 단체들이 함께 고안한 NIFF(Notation Interchange File Format)[2], 그리고 전자 음악 관련 단체들의 지원하에 제작된 MIDI (Musical Instrument Digital Interface)[13] 등이 있다. 표준 규격을 이용한 방법 외에도 특정 편집기를 통하여 음악 정보를 처리할 수 있으며 대표적인 편집기로 SCORE, Finale 등이 있다.

그러나 기존 방법들은 음악 정보에 대한 다양한 처리 방법을 제공함에도 불구하고 여러 가지 문제점을 포함하고 있다. SMDL의 경우 상당히 복잡한 SGML DTD로 정의되어 있으며, HyTime을 통한 멀티미디어 정보들의 링크와 동기화는 구현 과정이 매우 어렵다. 또한 SMDL을 지원하는 완벽한 어플리케이션이 제공되지 않고 있다[1]. NIFF는 인쇄된 악보를 단순히 음악 정보로 변환하기 위한 기능만을 제공하고 있으며 [2], MIDI는 음악 기호나 가사와 같은 정보를 표현할 수 없기 때문에 MIDI를 통한 악보의 표현은 어렵다[1, 8]. 또한 NIFF나 MIDI를 지원하는 시퀀서 프로그램들은 가격이 매우 고가이며 사용 방법도 복잡하기 때문에 일반 사용자들이 이용하기에 많은 어려움이 있다. 그리고 각각의 편집기에서 작성된 음악 정보는 서로 호환되지 않으며, 호환을 위해서는 별도의 변환기가 필요하다.

이러한 단점을 해결하기 위하여 웹 표준 언어인 XML (eXtensible Markup Language)[3, 4]을 음악 정보 표현에 이용하는 방법을 고안할 수 있다. 현재 XML을 이용하여 음악 정보를 처리한 기존 연구로는 MusicML [14]을 살펴볼 수 있다. MusicML은 XML을 이용하여 음악 정보에 대한 DTD(Document Type Definition)를 정의하여 이것을 자바 애플릿을 이용하여 악보로 출력한 것이다. 그러나 MusicML은 자바 애플릿을 이용하여 각각의 음악 기호를 처리하기 때문에 사용자가 내용을 변경하기가 어려우며, HTML 포맷을 사용하여 악보를 출력할 수 없기 때문에 기존 웹에서 사용하고 있는 관련 기술을 이용할 수 없다는 단점을 갖고 있다. 또한 현재 MusicML은 음악 정보를 악보로 출력하기 위한 출력 방법과 DTD만을 정의한 상태이며 그 결과 음악 정보에 대한 편집 및 재생 방법은 지원하지 못한다. 따라서 이러한 문제점들에 대한 개선이 필요하다.

본 논문에서는 이와 같은 문제점들을 개선하기 위하여 XML을 이용한 음악 편집 및 재생 시스템을 개발한다. 특별히 음악 정보 중 리듬 정보는 처리하기가 어렵기 때문에 본 연구에서는 리듬 처리를 다룬다. 음악 요소는 기본적으로 리듬(rhythm), 멜로디(melody), 하모니(harmony)로 구분할 수 있다. 이 중 리듬을 컴퓨터상에서 처리하기 위해서는 시공간적인 요소를 복합적으로 처리하여야 하기 때문에 단순한 음악 정보 처리보다 어려운 작업과정이 필요하다[10, 11]. 이러한 문제를 해결하기 위하여 본 논문에서는 RDML(Rhythm Description Markup Language)을 정의하였고, 정의한 리듬 정보의 표현은 기존의 매우 복잡한 음악 정보 표현 방법과는 달리 리듬 정보를 간결하게 표현하는 동시에 리듬의 비트, 악기의 종류, 출력 정보와 같은 리듬 정보를 정확하게 표현할 수 있다. 또한 본 논문에서는 RDML을 기본으로 하는 리듬 편집 및 재생 시스템을 구현한다. 본 시스템의 리듬 편집 및 출력, 그리고 재생 시스템은 웹 기반에서 수행된다. 기존 음악 정보 처리 방법들은 독자적인 파일 포맷을 이용하며, 일반 사용자들이 접하기 어려운 단점이 있으므로 이를 해결하고자 웹 기반의 시스템을 구현한 것이다.

2. 리듬 마크업 언어 RDML

XML을 이용한 리듬 정보 처리를 위하여 RDML (Rhythm Description Markup Language)을 정의하였다. RDML은 리듬 정보를 마크업 언어로 표현하기 위한 다양한 논리적 정보를 포함하고 있으며 기존 HTML 사용자들도 쉽게 사용할 수 있을 정도의 간결한 구조를 제공하고 있다. RDML DTD에서 표현하고 있는 중요 리듬 정보를 정리하면 다음 <표 1>과 같다. <표 1>에서처럼 RDML에서는 리듬 정보 처리를 위한 악보의 소절, 리듬, 여러 종류의 악기, 연음 처리, 그리고 리듬 정보가 동시에 발생할 경우 이를 처리하기 위한 경우를 논리적으로 정의하였다. 이러한 RDML DTD의 일부 내용은 다음 (그림 1)과 같으며 전체 내용은 부록으로 첨부한다.

〈표 1〉 RDML DTD에서 표현하고 있는 중요 리듬 정보

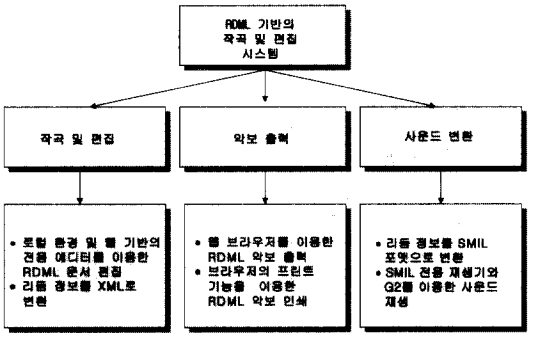
엘리먼트 이름	애트리뷰트	기능 설명
section	없음	악보의 소절
preinfo	rhythm	리듬의 박자
segment	seginfo	악보의 한마디
par	없음	동시 발생의 경우
hh	beat, status, info	HiHat 표현
snare	beat, info, inline, alone	Snare 표현
tom	beat, info, linline, kind	Tom 표현
kick	beat, info, linline	Kick Base 표현
symbol	beat, info	Symbol 표현
rest	length, position	쉼표
line	point, position, kind	연음 처리
vertical	size, repeat, ds, dc	DS, DC 처리

```
<!--attribute definitions-->
<!-- rhythm 3/4, 2/4, 6/8, 1/2, 4/4, 5/4 -->
<!ENTITY % att-preinfo
'rhythm(threequarter|twoquarter|sixeight|half|fourquarter|
fivequarter) #IMPLIED'>
<!--segment information for repeat, dal segno,
da capo-->
<!ENTITY % att-segment 'seginfo CDATA #IMPLIED'>
<!--HiHat beats,status,and information-->
<!ENTITY % att-hbeat
'beat(one|half|quarter|eighth|sixteenth|dotone|dothalf|
dotquarter |  doteighth) #IMPLIED'>
<!ENTITY % att-hstatus 'status (open|close) "close"'>
<!ENTITY % att-hinfo 'info (L|R|accent) #IMPLIED'>
<!ENTITY % att-hend 'end (y) #IMPLIED'>

  ----------      이하생략      ----------
```

(그림 1) RDML DTD의 일부

3. 리듬 정보 처리 시스템

RDML을 이용한 리듬 정보 처리를 위한 본 시스템의 구성도는 (그림 2)와 같다.



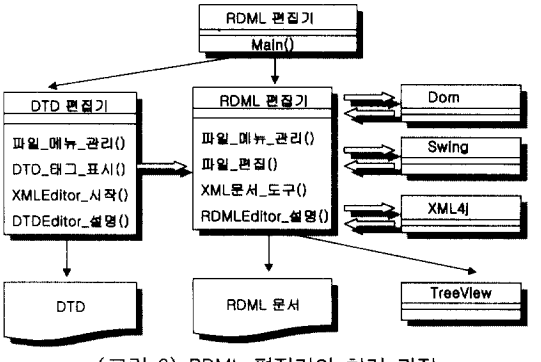
(그림 2) 리듬 정보 처리 시스템 구성도

3.1 편집기를 이용한 RDML 문서의 작성

RDML 문서 인스턴스를 작성하기 위해서는 RDML DTD에 정의된 엘리먼트 및 기타 태그들의 발생 순서 및 규칙들을 올바르게 사용하여야 한다. 본 논문에서는 이를 위하여 로컬 환경 및 웹 기반의 RDML 전용 편집기를 구현하였다.

3.1.1 로컬 환경의 RDML 편집기

로컬 환경의 자바 기반 RDML 편집기는 크게 DTD 편집과 RDML 문서 편집 부분으로 구성되어 있다. (그림 3)에서는 이러한 RDML 편집기의 처리 과정을 나타내고 있다.



(그림 3) RDML 편집기의 처리 과정

DTD 편집 처리 과정에서는 새로운 DTD를 작성하거나 이미 작성된 DTD를 읽어올 수 있다. 새로운 DTD를 작성할 경우에는 DTD 작성에 필요한 엘리먼트 및 애트리뷰트, 엔티티등을 메뉴 항목으로 입력할 수 있도록 하여 DTD 작성시 발생할 수 있는 오류를 방지하게 된다. 또한 기존의 DTD를 불러올 경우 DTD에 정의되어 있는 항목들을 XML 파서를 통하여 추출하게 되며 이 정보를 RDML 문서 작성시 메뉴 항목으로 제공하게 된다. 따라서 DTD에 정의된 논리적 정보들을 직접 입력하지 않고서도 간단하고도 올바르게 RDML 문서로 나타낼 수 있다.

DTD 편집 과정에서 추출된 정보에는 DTD에 정의된 엘리먼트 및 엔티티, 애트리뷰트 등이 포함되어 있다. 이 정보들은 RDML 문서 편집시 메뉴 항목으로 표현되며 사용자는 이 메뉴들을 이용하여 RDML 문서를 작성하게 된다. 따라서 잘못된 태그 구성이나 오타를 방지함으로써 정확한 문서 인스턴스를 만들 수 있다. 또한 문서 편집기를 통하여 생성된 RDML 문서는 트

리뷰를 통하여 논리적 정보를 시각적으로 표현함으로써 작성된 문서의 구조를 쉽게 이해할 수 있게 된다.

3.1.2 웹 기반의 RDML 편집기

RDML 편집기는 로컬 환경뿐만 아니라 웹 기반에서도 수행된다. 로컬 환경의 RDML 편집기는 매우 상세한 부분까지 RDML을 편집할 수 있지만 XML 문법이나 RDML DTD의 구성을 잘 모르더라도 쉽게 RDML 문서를 작성할 수 있도록 본 논문에서는 GUI를 지원하는 웹 기반의 RDML 편집기를 제공한다. 웹 기반의 RDML 편집기에서는 HTML의 입력 양식을 이용하여 RDML 문서를 작성하게 된다. 본 편집기의 입력 양식에는 RDML 문서 작성을 위한 모든 정보가 포함되어 있으며 한 트랙당 한 정보를 입력한다. 사용자가 입력한 트랙 정보는 RDML 문서로 변환되며 본 논문에서는 ASP의 객체를 이용하여 트랙당 선택된 입력 양식의 정보를 조합하였다. 그 과정은 다음 (그림 4)와 같다.

```
// CAFR(Conversion Algorithm : Form to RDML)
for track length N{  // n은 트랙의 길이
 if TAFR receive RDML information about rhythm instruments
    { // 입력양식의 리듬 악기 정보가 입력된 경우
     if TAFR receive "beat" information
       { // 비트 정보가 입력된 경우
        if TAFR receive "info" information
         { // info 양식의 정보가 입력된 경우
          write RDML element for "beat", "info" information
           //해당 RDML 태그 생성 }
        else if TAFR receive another information about "info"
               attribute
             { //info 양식의 정보가 새로운 속성일 경우
             write RDML element about "info" attribute
             //새로운 속성을 포함한 RDML 태그 생성 }
        else { //기타 입력 양식 정보가 없을 경우
             write RDML default element
              // 기본 태그 생성
           }
        }
      }
   }
```

(그림 4) 입력 양식에 대한 RDML 변환 알고리즘

3.2 웹 브라우저를 통한 RDML 문서의 출력

편집기를 통하여 작성된 리듬 정보는 악보로 출력할 수 있어야 한다. 본 논문에서는 RDML 문서를 출력하기 위하여 웹 브라우저를 사용하였다. 웹 브라우저를 사용함으로써 별도의 출력 어플리케이션 없이도 사용자가 간단히 악보를 출력할 수 있기 때문에 고가의 악보 출력 어플리케이션을 대체할 수 있다. 또한 네트워크상에서 웹 브라우저를 통한 악보의 표현 및 공유가 가능하다는 장점을 갖게 된다.

XML로 정의된 RDML 문서를 웹 브라우저에서 출력하기 위하여 본 논문에서는 RDML 문서를 HTML 문서로 변환하게 된다. 또한 악보 출력시 사용되는 해당 음악 기호를 VML(Vector Markup Language)[7]과 이미지를 사용하여 처리하기 때문에 이미지만을 이용한 출력 방법보다 세려한 출력 결과를 얻을 수 있다. 그 결과 본 논문의 악보 출력 방법은 MusicML의 애플릿을 통한 악보 출력 방식보다 간단한 처리 방법을 제공하며, 사용자가 내부 구조를 변경하기가 용이하다. 또한 HTML 기반의 출력 방식은 DHTML과 같은 다양한 웹 테크닉을 응용할 수 있으며, 쉽게 다른 환경에서도 공유할 수 있다.

3.2.1 DOM을 이용한 RDML 문서의 변환

DOM(Document Object Model)[6]이란 HTML과 XML 문서를 다루기 위한 W3C의 트리 기반 API 표준으로서 플랫폼 및 언어 중립적인 인터페이스를 제정한 것이다. DOM은 XML 문서를 객체화하여 이를 DOM 트리형태로 접근 할 수 있게 해준다.

```
//CARH(Conversion Algorithm : RDML to HTML)
function getchildren() // RDML문서의 HTML 출력 함수
{ var x=childnode of root node;  //자식 노드 생성
  var z=length of childnode // 자식노드의 길이

 if z is not ended  // 모든 자식노드의 길이 만큼 노드 탐색
 { for(index i, 0<= i < z)
  { if RDML has single instrument elements
   { // 해당 리듬이 동시에 발생하지 않을 경우
     if x(i) has RDML element
     { // 인덱스 x(i)에서 RDML 엘리먼트일 경우
      print current rhythm information
       // 해당 리듬을 악보로 출력
      getchildren( next node ) //다음 노드를 탐색
     }
     else if x(i) has another instrument elements
     {// 새로운 엘리먼트가 발생할 경우
      print current rhythm information
      // 해당 리듬을 악보로 출력
      getchildren( next node ) //다음 노드를 탐색
     }
   }
    else RDML has "par" element
      // 해당 리듬이 동시에 발생할 경우
     { if x(i) has RDML element
       {print current rhythm information
         // 해당 리듬을 악보로 출력
        getchildren( next node ) //다음 노드를 탐색
       }
      }
    }
   }
  }
```

(그림 5) RDML 문서의 악보 출력 알고리즘

따라서 XML 어플리케이션 개발이 특정 파서에 의존하지 않게 되며, 그 결과 DOM을 지원하지 않는 파서보다 XML 처리 방법이 용이하게 된다[6]. 본 논문에서는 RDML 문서를 HTML 문서로 변환하기 위하여 DOM 인터페이스를 통하여 RDML 문서 노드를 생성하여 이를 처리한다. 생성된 DOM 노드를 이용하여 RDML 문서를 HTML 문서로 변환하기 위해서는 적절한 트리 탐색기법이 필요하다. 이 과정을 본 논문에서는 (그림 5)에서와 같이 DFS 탐색 기법을 이용한 RDML 문서 변환 방법을 이용하여 RDML 엘리먼트를 HTML 태그로 변환하게 된다.

3.3 RDML 문서의 사운드 재생

RDML 리듬 정보는 악보의 출력 뿐만 아니라 사운드로 변환시켜 그 결과를 음으로 확인할 수 있어야만 완전한 작곡 시스템이라 할 수 있다. 따라서 본 연구에서도 RDML 문서를 SMIL(Synchronized Multimedia Integration Language)[5]을 이용하여 사운드로 재생하는 방법을 구현하였다. SMIL은 XML을 이용한 멀티미디어 데이터간의 동기 관계와 링크 방법을 지정할 수 있기 때문에 XML 음악 문서를 SMIL을 이용하여 시간적으로 동기화시켜 사운드로 출력하게 된다. SMIL을 통한 사운드 재생 방법을 통하여 사용자들은 네트워크 상에서 안정적으로 사운드를 출력할 수 있을 뿐만 아니라 로컬 환경에서도 간단하게 사운드를 재생할 수 있다.

3.3.1 DOM을 이용한 RDML 문서의 SMIL 변환

SMIL을 이용한 리듬 정보의 사운드 재생을 위해서는 RDML 문서를 SMIL 형식으로 변환하여야 한다.

본 연구에서는 이것을 DOM을 이용하여 SMIL 형식으로 변환하게 된다. 즉, RDML의 리듬 정보를 DOM 노드로 표현하여 해당 엘리먼트를 SMIL 태그로 변환하게 된다. (그림 6)에서는 앞서 설명한 SMIL변환 과정을 나타내고 있으며, RDML 문서의 HTML 변환 과정과 마찬가지로 DOM 노드를 생성하여 DFS 탐색을 이용한 것이다.

3.3.2 SMIL 문서의 사운드 재생

변환된 SMIL 문서는 RealPlayer의 G2나 CWI의 SMIL 재생기를 통하여 사운드로 확인할 수 있다. 그러나 현재 이러한 SMIL 재생기는 본 논문에서 제공하는 리듬 정보를 완벽히 사운드로 재생하지 못하고 있다. 그 이유는 SMIL 재생기가 사운드 전용이 아닌 기타 멀티미디어 요소들을 함께 처리하기 때문에 연속적인 리듬 음원 재생시 자주 끊기는 현상이 발생하기 때문이다.

```
//CARS(Conversion Algorithm : RDML to SMIL)
function getchildren() //  RDML 문서의 SMIL 변환 함수
{
var x=childnode of root node;  //자식 노드 생성
var z=length of childnode // 자식노드의 길이
if z is not ended  // 모든 자식노드의 길이 만큼 노드 탐색
 {
  for(index i, 0<= i < z)
  {
   if x(i) has RDML element
      { // 인덱스 x(i)에서 RDML 엘리먼트일 경우
      print SMIL element
            // 해당 SMIL 엘리먼트로 변환
        getchildren( next node ) //다음 노드를 탐색
      }
   elseif x(i) has another instrument elements
      { //새로운 엘리먼트가 발생할 경우
       print another SMIL element
       // 해당 SMIL 엘리먼트로 변환
       if x(i) has another attribute
       { // 속성을 포함한 엘리먼트가 발생 할 경우
         print SMIL element for another attribute
         getchildren( next node ) //다음 노드를 탐색
       }
         getchildren( next node ) //다음 노드를 탐색
      }
     }
  }
 }
```

(그림 6) RDML 문서의 SMIL 변환 알고리즘

따라서 본 연구에서는 SMIL 사운드 재생 성능을 향상시키기 위하여 별도의 SMIL 재생기를 제공한다. 사용자가 작성한 RDML 문서는 SMIL 변환 과정을 통하여 사운드 정보로 변환되며 이러한 사운드 정보를 출력하기 위해서는 앞서 설명한 변환 과정과 유사한 사운드 처리 과정을 거치게 된다. 즉 SMIL에서 표현하고 있는 사운드 정보를 DOM을 이용하여 추출하게 되며 각각의 경우에 적절한 사운드 파일을 출력한다. SMIL의 사운드 정보를 재생하기 위한 알고리즘은 (그림 7)과 같으며 본 재생기를 통하여 기존 G2나 CWI에서의 연속적인 음원 재생 결과보다 향상된 결과를 얻게 된다.

```
//SMIL Play Algorithm(SPA)
function getchildren() //SMIL 문서의 재생 알고리즘
{
 var x=childnode of root node;  //자식 노드 생성
 var z=length of childnode // 자식노드의 길이
if z is not ended  // 모든 자식노드의 길이 만큼 노드 탐색
 {
  for(index i, 0<= i < z)
  {
     if x(i) has "PAR" element
        { // PAR 엘리먼트가 발생할 경우
          getchildren( next node) //다음 노드를 탐색
        }
     else if x(i) has information about rhythm instrument
       { // 리듬악기 정보가 발생할 경우
         if x(i) has attribute about playing time
          { // 악기의 비트별로 재생 시간을 설정
            Play rhythm instrument // 해당 악기 음원 재생
            getchildren( next node ) //다음 노드를 탐색
          }
       }
   }
  }
 }
```

(그림 7) SMIL의 사운드 재생 알고리즘

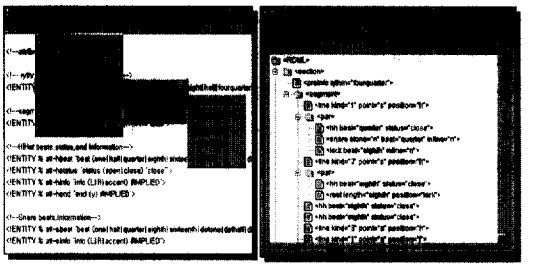
4. 시스템 구현

본 시스템은 RDML 문서의 편집 부분, RDML 리듬 정보의 악보 출력 부분 및 재생 부분으로 구성되어 있다. 이 시스템은 Windows/NT 4.0 운영체제에서 JDK 1.2와 IMB XML4J 파서 및 ASP(Active Server Pages)를 이용하여 편집기를 구현하였고, 악보 출력 및 재생을 위하여 마이크로소프트사의 XML 파서, EMAC DOM 스트림트, 인터넷 익스플로러 5.0, 그리고 Visual C++ 6.0을 이용하였다.

4.1 RDML 편집기

4.1.1 로컬 환경의 RDML 편집기

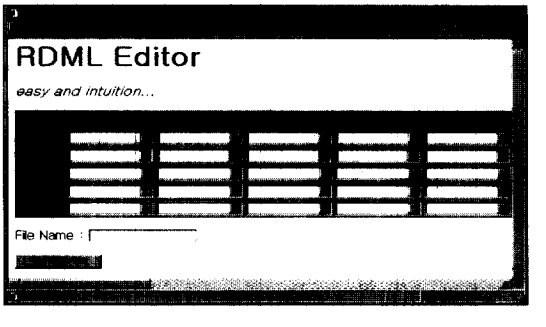
로컬 환경의 RDML 편집기는 DTD 작성 및 편집 모듈과 XML 문서 작성 및 편집 모듈, 그리고 XML 문서에 대한 Treeview 모듈로 구성되어 있다. DTD 작성 및 편집 모듈에는 기존 DTD 파일에 대한 입·출력 및 새로운 DTD 작성 기능을 포함하고 있다. XML 편집 모듈에서는 버퍼에 저장된 DTD 정보를 XML 파서를 통하여 엘리먼트 및 애트리뷰트 등에 관한 정보를 추출하게 되며, 추출된 정보를 메뉴항목으로 전달하여 이것을 화면에 출력 및 저장하게 된다. 그 결과 XML 편집 방법에서도 직접 입력 및 메뉴 항목을 통한 RDML 문서 작성이 가능하게 된다. (그림 8)에서는 지금까지 살펴본 로컬 환경의 RDML 편집기의 인터페이스를 나타내고 있다.



(그림 8) RDML 문서 편집기의 인터페이스

4.1.2 웹 기반의 RDML 문서 편집기

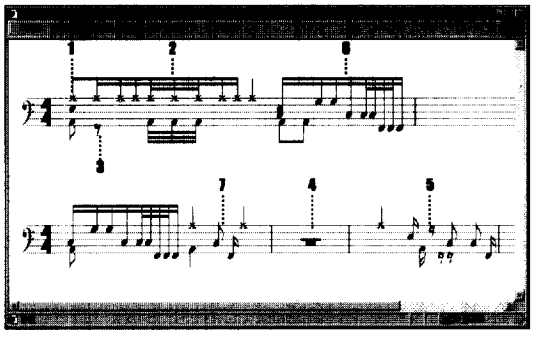
웹 기반의 RDML 편집기는 HTML의 입력 양식을 이용하여 리듬 정보를 표현하게 된다. 본 연구의 로컬 환경의 입력기는 DTD 정보 및 RDML 문서의 작성과 같은 세부적인 정보들을 처리할 수 있으나, XML이나 RDML의 DTD에 대한 지식이 부족한 일반 사용자들을 위하여 보다 간단하고 쉽게 RDML 문서를 작성할 수 있도록 웹 기반의 편집기를 제공한다. (그림 9)에서는 이러한 웹 기반의 문서 편집기 통하여 간단한 RDML 문서를 작성하는 과정을 나타내고 있다. 사용자는 각 트랙당 RDML의 기본정보, 리듬의 종류, 박자, 출력 위치 등을 선택하게 되며 최종적으로 자신이 저장하고자 하는 파일 이름을 입력한다. 그 결과 사용자로부터 선택된 입력 양식의 정보는 ASP를 통하여 RDML 문서로 저장되며 그 결과를 웹 브라우저에서 확인하게 된다.



(그림 9) 웹 기반의 RDML 편집기 인터페이스

4.2 웹 브라우저를 이용한 RDML 문서의 출력

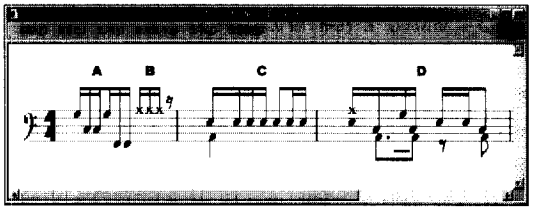
본 시스템에서 RDML 문서의 악보 출력은 앞서 살펴본 바와 같이 DOM을 이용한 HTML 문서로의 변환 과정을 통하여 수행된다. RDML 문서의 변환은 현재 DOM API를 지원하는 프로세서가 인터넷 익스플로러 5.0에 내장되어 있기 때문에 웹 브라우저를 이용한 것이다.



(그림 10) RDML 문서를 웹 브라우저에서 출력한 결과

RDML 문서를 DOM 스크립트를 이용하여 악보로 출력한 결과가 (그림 10)이며 각 사항을 세부적으로 설명한다.

- 리듬의 발생 순서 처리 : RDML에서의 리듬 발생 순서는 리듬 악기에 대한 태깅 순서이기 때문에 순차적인 표현 방식을 기본으로 하고 있다. 이러한 순차적 리듬 표현 결과를 (그림 10)의 (1)에서 보여주고 있다.
- 리듬의 연음 처리 : 리듬은 단독으로 발생할 수도 있지만 연속으로 처리해야 할 경우도 있다. 또한 상·하단부에 위치한 리듬 악기의 연음을 처리할 수 있어야 하며, RDML에서는 리듬의 비트에 따라 최대 3잇단음까지 표현하기 때문에 3단의 잇단음까지 표현할 수 있어야 한다. 그 결과 (그림 10)의 (2)부분에서와 같이 상·하단에 걸친 3잇단음이 처리될 수 있다.
- 쉼표 처리 : 쉼표는 리듬악기와 함께 순차적 혹은 동시에 발생할 수 있으며, 단독으로도 발생할 수 있다. 또한 쉼표는 리듬의 상하·좌우에 발생하기 때문에 쉼표에 대한 발생 위치를 지정할 수 있어야 한다. (그림 10)에서처럼 (3), (4), (5)에서와 같이 쉼표는 여러 위치에서 단독 혹은 다른 리듬과 함께 출력된다.
- 다양한 리듬 악기 처리 : RDML 악보에서는 TOM, HH, SNARE, KICK과 같은 다양한 리듬 악기를 표현한다. 이러한 다양한 리듬을 악보에서 구별하기 위해서는 각 리듬 악기에 모양과 출력 위치를 구분하여야 한다. (그림 11)의 (A)와 같이 TOM의 출력은 Large, Medium, Small TOM에 따라 오선 상단 첫마디, 세 번째 마디, 그리고 마지막 라인 하단부분에서 이루어진다. HH은 (B)와 같이 오선의 첫 라인에 걸쳐 출력되며, 다른 악기들과 출력 형태가 다름을 알 수 있다. SNARE 드럼은 (C)와 같이 오선 2번째 마디에서 출력되며, KICK Base Drum은 (C)부분의 첫 마디에서처럼 오선 4번째 마디에서 처리된다. 이러한 리듬악기 출력 방법을 이용하여 (그림 11)의 D부분과 (그림 10)의 (1),(6)(7) 부분에서처럼 다양한 리듬 악기들을 위치에 따라 구별하게 되며, 동시에 여러 리듬을 표현하게 된다.

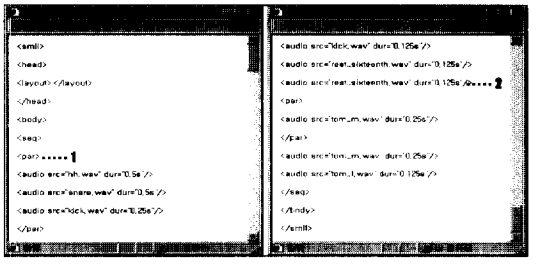


(그림 11) RDML에서 정의한 다양한 악기의 출력 결과

4.3 RDML 문서의 사운드 재생

4.3.1 RDML 문서의 SMIL 변환

RDML의 리듬 정보는 SMIL로 변환되며 이를 위하여 본 논문에서는 웹 브라우저 기반에서 DOM을 이용하였다. 그 결과 RDML의 리듬 정보는 시간적으로 동기화된 사운드 정보로 변환된다. DOM을 이용하여 RDML 문서를 SMIL 포맷으로 변환한 결과를 (그림 12)에서 나타내고 있으며, 세부적으로 살펴본다.

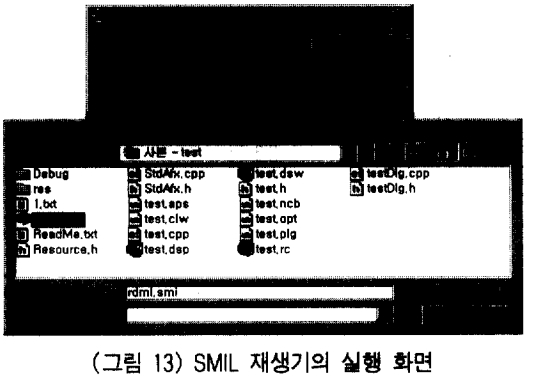


(그림 12) RDML 문서를 SMIL로 변환한 결과

- 리듬의 발생 순서 처리 : RDML에서 <par>로 처리된 리듬은 SMIL에서도 <par>를 이용하여 각 리듬을 동시에 재생하여야 하며, 기타 리듬들은 <seq>를 통하여 순차적으로 재생된다. 이 경우를 (그림 12)의 (1)에서처럼 <par> 혹은 <seq>를 이용하여 처리하게 된다.
- 리듬의 비트 처리 : RDML의 모든 리듬은 다양한 비트를 갖고 있다. 따라서 이러한 비트를 사운드로 처리하기 위해서는 비트에 따른 재생 시간을 SMIL에서 조절할 수 있어야 한다. 이를 위하여 본 논문에서는 SMIL의 "dur" 애트리뷰트를 이용하여 리듬의 비트별로 적절한 재생 시간을 지정하였다. 따라서 SMIL에서는 RDML의 다양한 비트를 적절한 사운드 재생 시간으로 표현하게 되며 그 결과는 (그림 12)의 (2)에서와 같다.

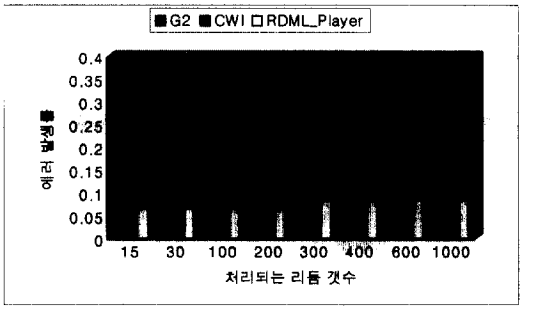
4.3.2 SMIL 문서의 사운드 재생

변환된 SMIL 문서는 현재 RealPlayer G2 또는 CWI에서 제작한 SMIL 전용 플레이어를 통하여 사운드로 재생할 수 있다. 그러나 기존의 SMIL 재생기는 사운드 전용 재생기가 아니기 때문에 연속적인 음원 재생시 자주 끊기는 단점이 있다. 따라서 본 연구에서는 (그림 13)과 같은 사운드 출력을 위한 SMIL 전용 재생기를 제공한다. 본 연구에서 개발한 재생기의 출력 성능과 G2 및 CWI의 재생기 성능을 비교한 결과는 (그림 14)와 같다. 그 결과 기존 SMIL 재생기는 처리할 리듬 개수와 성능이 반비례함을 알 수 있으며 본 연구에서 개발한 재생기를 통하여 보다 안정된 리듬 재생 결과를 얻을 수 있었다.



(그림 13) SMIL 재생기의 실행 화면

본 연구에서 개발한 SMIL 재생기는 먼저 사운드 출력을 위하여 파일을 선택한다. SMIL 파일이 선택된 후에는 바로 사운드로 출력되며, 사운드 출력을 위하여 wav 파일을 이용하였다.



(그림 14) SMIL 문서의 재생 성능 비교

5. 결 론

본 논문에서는 XML(eXtensible Markup Language)을 이용한 음악 편집 및 웹에서의 재생 시스템을 개발하였다. 특별히 음악 정보 중 리듬 정보는 처리하기가 어렵기 때문에 본 연구에서는 리듬 처리를 다루었다. XML을 이용한 음악 정보 처리를 위해서는 음악 표현을 위한 마크업 언어의 정의와 악보 편집 및 출력, 사운드로의 변환, 그리고 음악 정보에 대한 호환 등이 해결되어야 한다. 이를 위하여 본 연구에서는 RDML(Rhythm Description Markup Language)이라는 마크업 언어를 정의하였고, DOM과 VML을 이용하여 RDML 리듬 정보를 웹 브라우저에서 보다 쉽고 선명하게 출력할 수 있었다. 또한 리듬 정보를 SMIL 포맷의 사운드 정보로 변환하고 이를 SMIL 전용 재생기를 통하여 일반 사용자들도 쉽게 결과를 확인할 수 있도록 하였다. 그 결과 본 시스템에서는 누구나 쉽게 사용할 수 있는 리듬 정보 편집 환경과 웹에서의 출력, 사운드로의 변환 및 재생 기능을 제공하며, 본 연구에서 정의한 XML 포맷의 음악 정보는 어느 플랫폼에서도 완벽히 호환된다.

향후 완전한 음악 정보 처리를 위하여 본 연구에서 구현한 시스템에서 리듬 정보 이외의 일반적인 음악 정보 처리 방법과 시각적으로 개선된 편집 환경, 그리고 보다 향상된 음질을 제공하기 위한 연구가 진행중이다.

참 고 문 헌

[1] Commisson of the European Communities, "Report on SMDL Evaluation," http : //www.svb.nl/project/cantate/cantate.htm, 1999.

[2] Musitek, "Niff Notation File Standard," http : //www.musitek.com/niff.html, 1999.

[3] Charles F. Goldfarb and Paul Prescod, *The XML Handbook*, Prentice Hall, 1998.

[4] Tim Bray and Jean Paoli, "Extensible Markup Language(XML) 1.0," http : //www.w3c.org/TR/1998/REC1998210.html, 1999.

[5] Philipp Hoschka and Lynda Hardman, "The WWW Consortium issuses SMIL 1.0 as a W3C Recommendation," http : //www.w3c.org/Press/1998/SMIL-REC, 1998.

[6] Lauren Wood and Vidur Apparao, "Document Object Model(DOM) Level 1 Specification," http : //www.w3.org/TR/-REC DOM Level-1, 1999.

[7] Brian Mathews and Daniel Lee, "Vector Markup Language (VML) World Wide Web Consortium Note," http : //www.w3.org/TR/NOTE-VML, 1999.

[8] Tim Kientzle, *A Programmer's Guide To Sound*, Addison Wesley, 1998.

[9] B.H. Bronson, "Mechanical help in the study of folksong," Journal of American Folklore Vol.62, pp. 81-86, 1949.

[10] Alejandro Pazos and A. Santos del Riego, "Genetic Music Compositor," Proceedings of the 1999 Congress on Evolutionary Computation, Vol.2, No.3, pp. 885-890, 1999.

[11] D. V. Oppenheim, "DMIX : A Multifaceted Environment for Composing and Performing Computer Music," Computers Math, Vol.32, No.1, pp.117-135, 1996.

[12] Roger B. Dannenberg, *Music Understanding by Computer*, 1987/1988 Computer Science Research Review, School of Computer Science, Carnegie Mellon University, pp.19-28, 1987.

[13] The International MIDI Association, "MIDI 1.0 Detailed Specification," 1990.

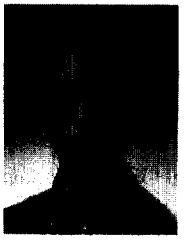
[14] Jeroen van Rotterdam, "MusicML an XML experience," http : //195.108.47.160/index.html, 1999.

[15] 이석호, 안영경, "음악 정보 처리 시스템의 설계 및 구현", 한국정보과학회지, Vol.9, No.4, pp.5-12, 19-82.

부록 RDML DTD

```
<!--attribute definitions-->
<!-- rythm 3/4, 2/4, 6/8, 1/2, 4/4, 5/4 -->
<!ENTITY % att-preinfo 'rythm
(threequarter|twoquarter|sixeight|half|fourquarter|fivequar
ter) #IMPLIED'>

<!--segment information for repeat, dal segno, da capo-->
<!ENTITY % att-segment 'seginfo CDATA #IMPLIED'>

<!--HiHat beats,status,and information-->
<!ENTITY % att-hbeat 'beat
(one|half|quarter|eighth|sixteenth|dotone|dothalf|dotquarte
r|doteighth) #IMPLIED'>
<!ENTITY % att-hstatus 'status (open|close) "close"'>
<!ENTITY % att-hinfo 'info (L|R|accent) #IMPLIED'>
<!ENTITY % att-hend 'end (y) #IMPLIED'>

<!--Snare beats,information-->
<!ENTITY % att-sbeat 'beat
(one|half|quarter|eighth|sixteenth|dotone|dothalf|dotquarte
r|doteighth) #IMPLIED'>
<!ENTITY % att-sinfo 'info (L|R|accent) #IMPLIED'>
<!ENTITY % att-inline 'inline (y|n) "n"'>
<!ENTITY % att-alone 'alone (y|n) "n"'>

<!--Tom beats, kind and information-->
<!ENTITY % att-tbeat 'beat
(one|half|quarter|eighth|sixteenth|dotone|dothalf|dotquarte
r|doteighth) #IMPLIED'>
<!ENTITY % att-tkind 'kind (h|m|l) "m"'>
<!ENTITY % att-tinfo 'info (L|R|accent) #IMPLIED'>
<!ENTITY % att-inline 'inline (y|n) "n"'>

<!--Kick beats, information-->
<!ENTITY % att-kbeat 'beat
(one|half|quarter|eighth|sixteenth|dotone|dothalf|dotquarte
r|doteighth) #IMPLIED'>
<!ENTITY % att-kinfo 'info (accent) #IMPLIED'>
<!ENTITY % att-inline 'inline (y|n) "n"'>

<!--Symbol beats, information-->
<!ENTITY % att-sybeat 'beat
(one|half|quarter|eighth|sixteenth|dotone|dothalf|dotquarte
r|doteighth) #IMPLIED'>
<!ENTITY % att-syinfo 'info (accent) #IMPLIED'>

<!--Rest information-->
<!ENTITY % att-rest 'length
(one|half|quarter|eighth|sixteenth|dotone|dothalf|dotquarte
r|doteighth) #IMPLIED'>
```

```
<!ENTITY % att-rposition 'position (hh|kick|snare|th|tm|tl)
#IMPLIED'>

<!--Line information-->
<!ENTITY % att-point 'point (s|e) #IMPLIED'>
<!ENTITY % att-position 'position (h|l) #IMPLIED'>
<!ENTITY % att-kind 'kind CDATA #IMPLIED'>

<!--Vertical Line information-->
<!ENTITY % att-vsize 'size (one|double|end) #IMPLIED'>
<!ENTITY % att-repeat 'repeat (start|end) #IMPLIED'>
<!ENTITY % att-ds 'ds (start|jump|end) #IMPLIED'>

<!ENTITY % att-dc 'dc (start|jump|end) #IMPLIED'>

<!ELEMENT RDML     (section+)>

<!ELEMENT section  (vertical?,preinfo,(segment*))>

<!ELEMENT preinfo  EMPTY>
<!ATTLIST preinfo  %att-preinfo;>

<!ELEMENT segment
((par|hh|snare|tom|kick|symbol|rest|equal|line)*,vertical)>
<!ATTLIST segment  %att-segment;>

<!ELEMENT par      (hh|snare|tom|kick|symbol|rest|line)*>

<!ELEMENT hh       EMPTY>
<!ATTLIST hh       %att-hbeat;
                       %att-hstatus;
                       %att-hinfo;
                   %att-hend;>

<!ELEMENT snare    EMPTY>
<!ATTLIST snare    %att-sbeat;
                       %att-sinfo;
                       %att-inline;
                       %att-alone;>

<!ELEMENT tom      EMPTY>
<!ATTLIST tom      %att-tbeat;
                   %att-tkind;
                   %att-tinfo;
                   %att-inline;>

<!ELEMENT kick     EMPTY>
<!ATTLIST kick     %att-kbeat;
                   %att-kinfo;
                   %att-inline;>

<!ELEMENT symbol   EMPTY>
<!ATTLIST symbol   %att-sybeat;
                   %att-syinfo;>

<!ELEMENT rest     EMPTY>
<!ATTLIST rest     %att-rest;
                   %att-rposition;>

<!ELEMENT equal    EMPTY>

<!ELEMENT vertical EMPTY>
<!ATTLIST vertical %att-repeat;
                   %att-vsize;
                   %att-dc;
                   %att-ds;>

<!ELEMENT line ANY>
<!ATTLIST line %att-point;
               %att-kind;
               %att-position;>
```

손 원 성

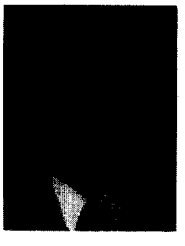
e-mail : stratus@thrunet.com

1998년 동국대학교 컴퓨터공학과 (공학사)

2000년 동국대학교 컴퓨터공학과 (공학석사)

2000년~현재 (주)사이버테크 연구원

관심분야 : XML, 컴퓨터 음악, 멀티미디어 정보처리



이 용 규

e-mail : yklee@dgu.ac.kr

1986년 동국대학교 전자계산학과 (공학사)

1988년 한국과학기술원 전산학과 (공학석사)

1996년 미국 Syracuse University 전산학 박사

1997년~현재 동국대학교 컴퓨터·멀티미디어공학과 교수

관심분야 : 멀티미디어 저장 시스템, 하이퍼미디어 시스템, 데이터베이스, 정보검색