Journal of Korea Multimedia Society Vol. 7, No. 9, September 2004(pp. 1304-1311)

객체지향적 분석에 따른 SMIL 구조변환 및 모델링

권 오 현†

요 약

SMIL(Synchronized Multimedia Integration Language)은 각종 멀티미디어 데이터들 간의 동기화를 지원해 주며 데이터 요소들의 화면 구성을 편리하게 해줌으로써 인터넷 환경의 가상강의, 방송 등에서 널리 활용되고 있다. 응용분야가 다양하게 확대됨에 따라 각각의 환경에 융통성 있게 적용할 수 있는 체계의 필요성이 증대되고 있는 추세이며 이에 따라 SMIL 기능도 계속적으로 진화되고 있다. 본 논문에서는 SMIL의 주요 기능을 객체 지향적 방법론에 따라 분석, 모델링 한 후 서비스 컴비네이터(service combinator) 개념의 클래스 모델을 제안하여 설계함으로써 필요시 사용자들이 자기 환경에 맞게 튜닝 할 수 있도록 하였다.

Enhancement of SMIL by Changing Structure and Modeling Depending Upon Object Oriented Analysis

Oh-hyun Kwon†

ABSTRACT

SMIL supports synchronization and nice screen form environments among various multimedia data. Thus it is utilized widely in the fields of internet virtual lecture, internet broadcasting, and so on. Those kinds of system which can be applied flexibly in spite of the change of environments are required continuously with important issue, and depending upon the trends the function of SMIL is upgraded step by step. In this paper main function of SMIL is analyzed and modeled depending upon object oriented methodology. Also I propose some class model which is inspired from service combinator. Suggested class model can be tuned by user easily if he wants to optimize system environments.

Key words: Service Combinator(서비스 컴비네이터), UML(유엠엘), SMIL(스밀), Modeling(모델링), Class(클래스), Sequential Diagram(순서 다이어그램), WebL(웹블)

1. 서 론

분산 멀티미디어 환경에서 각종 다양한 미디어 데이터를 신뢰성 있게 송수신하기 위해서는 각기 다른 서버환경과 네트워크 환경의 이질성을 극복하기 위한 조치를 취해야 한다.

특히, 인터넷과 같은 네트워크 환경은 각종 멀티미디어 데이터 대역폭의 차이 및 속도의 차이 등으로 인해, 사용자가 요구하는 속도와 신뢰성을 갖춘 데이터를 지원하기가 어렵다. SMIL(Synchronized Multimedia Integration Language)[1] 환경에서는, 송수신시의 동기화 문제를 극복하기위해 송수신 속도를 설정 할 수 있도록 함으로써, 사용자차원 편리성을 강화하였으나, 고장시의 신뢰성을 극복하기에는 기능적으로 다소 부족하다.

Luca Cardelli와 Rowan Davies는 멀티미디어 환경의 신뢰성 보강을 위해 웹 환경에서의 서비스 컴비네이터(Service Combinator)[2]를 제안하였다. 그것은 동시성제어(concurrent access), 순차적 접근(sequential access), 시간제한(time limitation), 송수

※ 교신저자(Corresponding Author) : 권오현, 주소 : 부산광역시 남구 용당동(608-711), 전화 : 051) 610-8415, FAX : 051) 610-8039, E-mail : ohkwon@tit.ac.kr
접수일 : 2003년 10월 20일, 완료일 : 2004년 2월 17일
† 종신회원, 동명정보대학교 컴퓨터공학과
※ 본 연구는 2003년도 동명정보대학교 교내연구비 지원 과제임.

신 속도 제한(transmission rate limitation)등의 제어를 행할 경우 웨블(WebL) 같은 별도의 특수하게 개발된 웹 전용 언어를 활용하기 때문에 SMIL에 비해 상대적으로 신축성있게 제어가 가능하다는 이점이 있다. 본 논문에서는 주요 SMIL 기능을 대상으로 객체 지향적 방법론에 따라 분석을 한 후, 사용자가 필요시 수시 수정이 가능토록 서비스 컴비네이터와 유사하면서도 자바환경에서 구현이 가능한 클래스 모듈을 설계하였다.

2. 관련 연구

2.1 서비스 컴비네이터(Service Combinator)

서비스 컴비네이터는 WebL이 분산 환경에서 동작 될 때 다소 취약한 신뢰성을 보강시켜 줄 수 있다. 이 언어는 웹을 통한 정보교환을 보다 신뢰성있게 구현하는데 주안점을 두고 여러 가지 예외처리 관련 모듈들이 사용된다. 예를 들면, 정보교환에 실패할 수 있는 여러 요인들인 비정상적인 데이터 전송속도, 타임아웃(timeout), 재송신(retry), 병행처리(parallel processing) 등에 관한 서비스 모듈들을 수행할 때 웹 서버가 고장 났을 경우나 웹 페이지가 소실될 경우에 대비하여 throw an exception 등 보강된 예외처리 방식을 적절히 활용함으로써, 웹과 같이 각종 다양한 이기종 자원 환경에서 작업이 이루어지는 경우에 발생하기 쉬운 신뢰성 문제를 개선할 수가 있다.

2.2 웨블(WebL)

WebL[3]은 웹상의 각종 유형별 데이터들을 보다 자동화된 기법으로 처리하기 위해 개발된 언어로, 크게 서비스 컴비네이터 개념의 구조와, 마크업 대수(markup algebra) 개념의 구조를 복합적으로 가진 언어이며, 컴팩이 개발하여 웹상의 여러 가지 유형의 멀티미디어 데이터를 융통성있게 처리하는 용도로 활용을 하였다.

마크업 대수 기법은 웹 문서 페이지를 효과적으로 추출하는데 주로 사용되고 있으며, 크게 오퍼레이터와 태그(tag), 조각(piece) 및 조각세트(piece set)등 3가지 기능으로 구성되어 있다[4]. 태그는 <> 기호로 둘러싸인 문자들을 의미하며 조각은 시작과 끝을 의미하는 부호로 둘러싸인 부분으로 XML의 엘리먼트와 유사하며 파싱을 위한 기본 단위이다. 웨블 언어는 이처럼 서비스 컴비네이터를 활용하여 멀티미디어 데이터의 처리시의 신뢰성을 강화하는 동시에 웹 다큐먼트의 체계적 추출을 위해 마크업 대수 기법을 활용한다.

2.3 SMIL(Synchronized Multimedia Integration Language)

SMIL은 멀티미디어 요소들(이미지, 오디오, 비디오 등)간의 동기화를 지원해 주며, 데이터 요소들에 대한 화면 배치를 사용자가 쉽게 할 수 있도록 지원해 주는 언어로서 XML과 유사한 형태의 문법체계를 가지고 있다. 웹 에서 미디어들의 시간관계와 공간관계를 표현할 수 있고 다양한 미디어들을 대상으로 한다는 점에서 가상 강의, 인터넷 방송 등에 널리 활용될 수 있는 가능성을 제시하였다[5]. SMIL의 구성은 크게 머리 부분인 <HEAD> 요소와 몸체 부분인 <BODY> 요소로 나뉜다.

<HEAD> 요소는 프리젠테이션 할 미디어의 공간적인 배치 정보를 지정하는 레이아웃 요소와 SMIL 문서에 대한 메타정보를 나타내는 META 요소로 구분되며 <BODY> 요소는 미디어들 간의 동기화를 위한 동기화 요소와 미디어들의 시, 공간 정보를 기술하는 미디어 요소로 구성된다[6].

3. 재설계 대상 SMIL

위에서 살펴본 바와 같이 SMIL은 멀티미디어 데이터들을 사용자 환경에 적절하게 활용할 수 있는 여러 가지 기능들을 제공하며, 사용하기가 간편하다는 장점이 있는 반면에 사용자 입장에서 더욱 다양한 기능을 필요로 할 경우에 소스를 직접 접할 수 없기 때문에, 이의 수정이 어렵고, 다양한 이기종 들로 이루어진 웹 환경에서는 멀티미디어 데이터들 간의 상호 연동 시 일부 시스템의 고장 및 성능저하에 따른 신뢰성의 제한이라는 단점도 내포하고 있다[7].

본 연구에서는 이러한 점을 감안하여 서비스 컴비네이터의 개념을 응용한 객체 지향적 클래스 모듈을 개발하여 신뢰성을 강화시키고 일부 SMIL 기능을 보완코자 하며, 각 모듈의 명칭을 액세스 컴비네이터(access combinator)로 하였다. 세부적인 기능으로는 순서적 접근(sequential access), 동시 병행적 접근(concurrent access), 처리시의 시간적 제한(time-

limitation), 전송처리시의 전송속도제한(transmission rate limitation), 우선순위(priority), 반복처리(repetition) 등이 그 대상으로 포함된다[8].

Sequential access

```
<body>
<seq-access>
  <m1>
  <m2>
</seq-access>
</body>
```

기존 SMIL의 <seq> 명령문에서는 통상적으로 액세스할 대상 미디어 클립 <m1>이 아무 이상없이 정상적으로 끝났을 경우에 한해 <m2>를 처리하는데 비해 새로 추가시킨 <seq-access> 컴비네이터는 대상 미디어 클립중 하나만 성공해도 정상적인 처리를 계속할 수 있도록 하였다.

<seq-access> 컴비네이터는 액세스할 대상 클립 <m1>을 처리한 후 이어서 <m2>를 처리하는데 만일 <m1>이 고장이 났을 경우는 관련 메시지와 함께 <m2>만을 처리하게 되며 그 반대의 경우는 <m1>만을 처리하게 된다.

작업 중인 클립이 실패하였는지 여부는 time limitation을 설정하여 그 시간을 초과했을 경우 실패한 것으로 간주한다.

Concurrent access

```
<body>
<par-access>
  <m1>
  <m2>
</par-access>
</body>
```

기존 SMIL의 <par> 명령문은 <m1> <m2>중 어느 하나만 실패해도 모두 실패한 것으로 처리토록 하는 것이 일반적이며 이 경우 제어를 단순하게 할 수 있다는 장점은 있으나 에러에도 불구하고 중단없이 지속적으로 처리하는 측면에서는 다소 취약하다. 이에 반해 새로 추가시킨 <par-access> 컴비네이터는 여러 개의 미디어클립들에 대한 액세스 <m1>과 <m2>중 하나만 성공해도 어떤 것이든 먼저 성공하게 되는 액세스의 결과가 리턴되며 실패한 클립에 대한 에러 메시지를 포함시켜 결과를 알려 준다. <m1>과 <m2>모두 실패하게 될 경우만 이 컴비네이터 역시 실패한 것으로 처리된다.

Time limitation

```
<body>
<time-limit value="t">
  <m>
</time-limit>
</body>
```

Time limitation 컴비네이터는 클립을 액세스하여 처리하는 시간제한을 두어 주어진 시간(=t)내에 완료되지 못할 경우는 실패한 것으로 간주하여 처리를 중단하는 기능을 수행하며 기존 SMIL의 duration time 설정과 유사하다.

Transmission rate limitation

```
<body>
<rate-limit value="t">
  <m>
</rate-limit>
</body>
```

Transmission rate limitation 컴비네이터는 전송속도가 <t>bps이하일 경우는 실패한 것으로 간주하게 되며 기존 <SMIL>에는 없는 기능을 추가로 설정하였다. 그 이유는 고속의 전송속도를 요구하는 미디어를 대상으로 옵션 기능으로 활용시 효과가 상대적으로 클 것으로 판단되기 때문이다.

Repetition

```
<body>
<repeat>
  <m>
</repeat>
</body>
```

Repetition 컴비네이터는 성공할 때 까지 계속 반복을 하게 하는 명령문으로 기존 SMIL의 명령문과 같다.

이상과 같은 몇 가지 수정된 제어문과 기존의 제어문을 복합적으로 활용할 경우 다음과 같은 명령문들을 생성시킬 수 있다.

(명령문 유형1) 5초 이내에 아무런 반응이 없으면 비디오 대신 이미지 클립을 동작

```
<body>
 <seq-access>
     <time-limit value="5sec">
         <video src="exam.mpeg"/>
     </time-limit>
     <image src="exam.jpg"/>
 </seq-access>
</body>
```

(명령문 유형2) 서버 A, B, C에서 비디오클립을 선택하되 만일 전송속도가 100Kbps 이하이면 서버 D에 있는 오디오파일을 동작

```
<body>
 <seq-access>
      <rate-limit value="100kbps">
          <par-access>
            <video src="A/exam.mpg"/
            <video src="B/exam.mpg"/
            <video src="C/exam.mpg"/>
          </par-access>
      </rate-limit>
      <audio src="D/exam.au"/>
  </seq-access>
 </body>
```

4. 변환 클래스 모듈의 모델링

SMIL 변환 클래스 모듈의 모델링 결과는 그림 1과 같다.

먼저 Access 클래스는 추상화 클래스로 설정하였다. 각각의 액세스 컴비네이터는 Access 클래스로부터 상속을 받으며, GetURL() 메소드는 작업 대상 자원의 처리를 요청하며 url() 메소드는 작업결과를 리턴 받는다.

Time-Limit 클래스는 미디어 객체를 접근 시 시간제한 값을 검사 및 처리하기 위해 사용되며, Rate-Limit 클래스는 미디어 객체를 전송시의 전송속도 제한 값과 시간제한 값을 검사 및 처리하기 위해 사용된다.

앞서 언급한 바와 같이 Seq-Access 클래스와 Par-Access 클래스는 미디어 데이터 클럽들의 순서적 처리와 동시 병행 처리를 위해 사용된다.

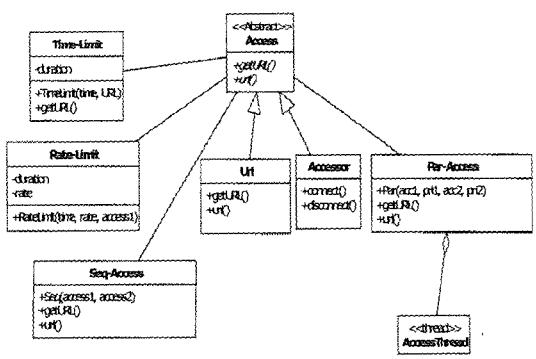

그림 1. 액세스 컴비네이터 클래스 다이어그램

액세스 컴비네이터의 주요 기능은 getURL() 메소드의 호출 순서에 따라 많은 영향을 받는다.

URL 클래스와 이와 관련한 메소드에 관한 알고리즘은 그림 2와 같다.

URL 객체를 생성시 생성자에 잘못된 URL 값이 넘겨질 수 있는 경우에 대비하여 예외처리 기능인 MalformedURLException을 활용토록 하였다. url() 메소드는 서버에서 작업한 결과를 리턴시키는 역할을 수행하며, 잘못된 결과로 인한 오류에 대비하여 try-catch 예외처리로 해결토록 하였다.

```
class URL // creation of URL object
 { public URL(String u) throws
   MalformedURLException
   public URL(String protocol, String
   host, String file) throws
   MalformedURLException
   public URL(String protocol, String
   host, int port, String file)
   throws MalformedURLException

url(String strURL) // get the resource
                      from server
{
 try {
  myURL= new URL(strURL);
  myCon.connect();
  is = myURL.openStream();
  br = new BufferedReader(new InputStr
       eamReader(is));
  while ((data = br.readLine()) != null)
  {
          System.out.println(data);
        }
       }
catch (MalformedURLException e) {}
catch(IOException e) {}
catch(Exception e) {}
}

getUrl() // request information of the
            resource

try{
URL u = new URL("resource");
u.getPort();
u.getfile();//get the location of file
}
catch (MulformedURLException e) {}
}
```

그림 2. URL 객체 알고리즘

geturl() 메소드는 클라이언트가 서버에게 작업을 요청하는 절차를 수행하며 역시 예외처리 기능을 포함시켰다.

각각의 액세스 컴비네이터 별로 이벤트가 이루어지는 수순은 순서 다이어그램에 의해 표현이 가능하다. Time limitation에 관한 작업 절차는 그림 3에서 보여지며, 전송 속도와 관련된 Transmission rate limitation은 그림 4, Sequential 액세스와 Concurrent 액세스는 그림 6과 그림 7에서 보여지고 있다.

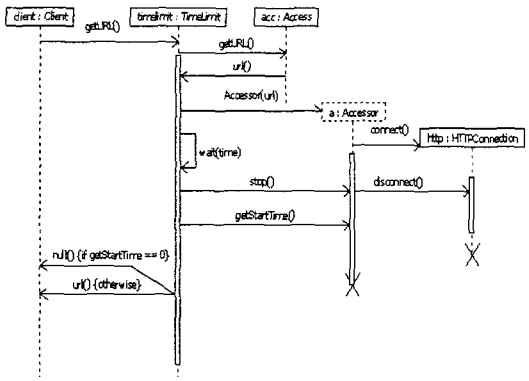

그림 3. Time limitation 순서 다이어그램

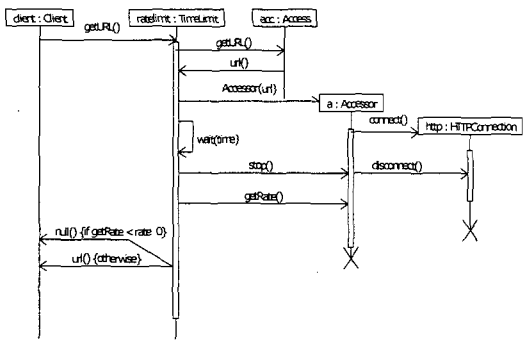

그림 4. Transmission rate limitation 순서 다이어그램

그림 3과 그림 4는 유사한 동작 특성을 보이나 다소 차이가 있다.

그림 3은 클립 처리시의 전체적인 시간 경과제한 값의 초과여부를 검사하기 위해 사용되는 반면, 그림 4의 경우 전체적인 시간 경과 뿐만 아니라 전송속도가 요구되는 제한 값에 맞지 않는지 여부를 검사한다. 즉, 클라이언트가 자원의 처리를 요구하면, ratelimit 객체는 그 요구사항을 수신하여 Accessor 객체에게 전달한다. 이때 Accessor 객체는 다시 HTTPConnection 객체에 연결을 시도하고 일정시간이 지나도록 연결이 안 되면 접속시도를 중단한다.

정상적으로 접속이 되고 전송속도가 설정된 최소 제한 값보다 크면 정상적인 결과가 클라이언트에게 전달되며 그 반대의 경우에는 실패의 결과 값인 null()이 전달된다.

Time limitation과 Transmission rate limitation의 공통부분인 wait(time) 산정을 위한 알고리즘은 아래 그림 5와 같다.

그림 6의 Sequential access 순서도에서 만일 클라이언트가 seq-access를 요청하면 첫 번째 객체의 주소를 접근하여 결과를 클라이언트에게 돌려준 후 이어서 두 번째 객체를 접근하게 된다. 만일 첫 번째 객체의 결과 값이 null()일 경우에도 두 번째 객체로의 접근이 시도되게 된다.

이것은 3장에서 언급한 바와 같이 기존의 SMIL 절차와 차이를 보이는 부분으로 고장에 따른 영향을

```
wait(time)
{
long tm=System.currentTimeMillis();
while (true)
{
  perform <m>;
  try {
  tm += time;
     Thread.sleep(Math.max(0,tm-System.
       currentTimeMillis())); //confirm the wait
       time
  } catch (InterruptedException e) {
  break;
  }
}
}
```

그림 5. wait(time) 산정 알고리즘

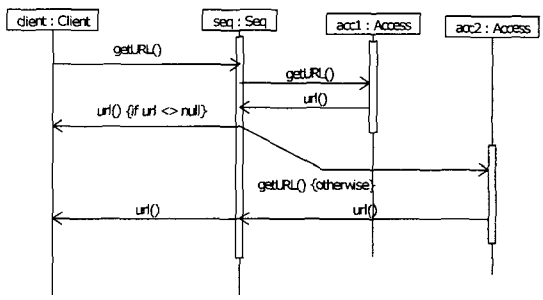

그림 6. Sequential access 순서다이어그램

최소화 시키는 역할을 수행한다.

그림 7의 Concurrent access 순서도를 살펴보면 클라이언트가 par-access를 요청했을 경우 필요한 만큼의 스레드들이 동시에 생성된다.

각각의 스레드들 중 성공하는 첫 번째 스레드에게 필요한 자원을 접근할 수 있는 권한을 부여하게 되며 그 결과 값은 클라이언트에게 리턴된다.

Par-Access의 알고리즘은 그림 8과 같다.

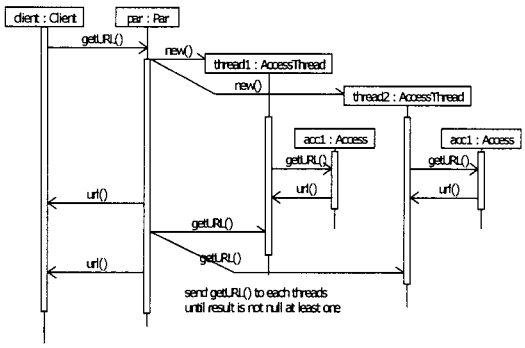

그림 7. Concurrent access 순서다이어그램

```
public class AccessThread extends Thread {
AccessThread thread1 = new
AccessThread();//creation of new thread
AccessThread thread2 = new
AccessThread();
thread1.start();
// execute run() of thread1 →
 thread1.getUrl() and url()
 thread2.start();
// execute run() of thread2 →
 thread2.getUrl() and url()
}
```

그림 8. Par-Access 알고리즘

5. SMIL 명령문의 재구성

설계된 변환 클래스 모듈들을 적용키 위한 SMIL 명령문들은 아래 그림 9와 같이 수정되어 적용된다.

헤더(header)부분은 원래의 명령문들을 그대로 사용가능토록 설계되었으며 몸체(body) 부분 중 수정이 필요한 명령부분 중심으로 제시하였다. par-access와 seq-access는 기존의 par, seq 명령문과 유사하나 고장시 지속적 처리가 필요한 경우를 위해 추가시켰으며 time-limit는 dur 어트리뷰트와 유사[9]하나 태그 개념으로 조정하여 추가시켰으며 rate-limit 명령은 전송속도의 최저치 설정 명령문의 개념으로 제안하였다.

```
<!-- The Document Body →
<!ENTITY % media-object "audio|video|
  text|img|animation|textstream|ref">
<!ENTITY % access-combinator "par-access|
  seq-access|time-limit|rate-limit|repeat|
  fail">
<!ENTITY % schedule "par|seq|(%media-obj
  ect;)|(%access-combinator)">
<!ENTITY % inline-link "a">
<!ENTITY % assoc-link "anchor">
<!ENTITY % link "%inline-link;">
<!ENTITY % container-content "(%sched
  ule;)|switch|(%link;)">
<!ENTITY % body-content "(%container-con
  tent;)">

<!ELEMENT body (%body-content;)*>
<!ATTLIST body %id-attr;>

<!-- Synchronization Attributes -->
<!ENTITY % sync-attributes "
  begin CDATA
     #IMPLIED
  end CDATA
     #IMPLIED
">

<!-- Switch Parameter Attributes →
<!ENTITY % system-attribute "
       system-bitrate CDATA
              #IMPLIED
       system-language CDATA
              #IMPLIED
       system-required NMTOKEN
              #IMPLIED
       system-screen-size CDATA
              #IMPLIED
       system-screen-depth CDATA
              #IMPLIED
       system-captions(on|off)
              #IMPLIED
       system-overdub-or-caption
            (caption|overdub)
              #IMPLIED
">

<!-- Fill Attribute →
<!ENTITY % fill-attribute "
        fill (remove|freeze) 'remove'
">

<!-- The Parallel Element →
<!ENTITY % par-content "%container-
content;">
<!ELEMENT par (%par-content;)*>
<!ATTLIST par
    %id-attr;
    %desc-attr;
```

```
    endsync        CDATA        "last"
    dur            CDATA        #IMPLIED
    repeat         CDATA        "1"
    region         IDREF        #IMPLIED
  %sync-attributes;
  %system-attribute;
>

<!-- The Sequential Element →
<!ENTITY % seq-content "%container-
con tent;">
<!ELEMENT seq (%seq-content;)*>
<!ATTLIST seq
 %id-attr;
 %desc-attr;
 dur            CDATA        #IMPLIED
 repeat         CDATA        "1"
 region         IDREF        #IMPLIED
 %sync-attributes;
 %system-attribute;
>

<!-- ACCESS COMBINATORS →
<!ENTITY % combinator-content
"(%access-combinator;)|
 (%media-object;)">
<!ELEMENT par-access
 (%combinator-content;)*>
<!ELEMENT seq-access
 (%combinator-content;)*>
<!ELEMENT time-limit
 (%combinator-content;)*>
<!ATTLIST time-limit
 dur CDATA        #REQUIRED>
<!ELEMENT rate-limit
 (%combinator-content;)*>
<!ATTLIST rate-limit
 dur CDATA        #REQUIRED
 time CDATA       #REQUIRED
>
<!ELEMENT repeat (%combinator-content;)*>
<!ELEMENT fail EMPTY>
```

그림 9. 수정된 SMIL 명령문

6. 기존 SMIL과의 비교분석

본 연구에서는 SMIL중 비교적 자주 사용되는 명령문을 대상으로 일부 기능을 추가 하였으며 기존의 기능과 비교를 하면 표 1과 같다.

표 1에서 순서적 처리 명령부분과 동시처리명령부분의 경우 추가된 명령문은 기존 명령문에 비해 고장에 대한 감내도는 양호하다 할 수 있다. 반면에 사용자의 요구사양 자체가 모든 대상 클립들에 대한 완벽한 동작을 전제로 할 경우에는 사용키 어렵다는 제한점이 있다. 즉, All or Nothing 전략을 채택할

표 1. 기존 SMIL 명령문과의 비교

기존 유사 명령문	추가 명령문	비 교
<seq>	<seq-access>	-기존 : 처리순서가 빠른 <m1>이 실패하면 전체가 실패한 것으로 간주 -추가 : 최소한 1개의 클립만 성공해도 작업 계속
<par>	<par-access>	-기존 : 작업 대상 클립 <m1> <m2>를 패키지 개념으로 설정하여 하나만 고장 나도 전체가 실패한 것으로 간주 -추가 : <m1><m2>중 하나만 성공해도 작업계속
없음	<rate-limit>	비교적 고속을 요하는 미디어 처리의 효율증대를 위해 추가

경우에는 추가된 명령문의 사용이 적합하지 않다. 또한 전송속도와 관련한 <rate-limit> 명령문은 고속을 요하는 특정 미디어를 대상으로 할 경우는 효과적이라 할 수 있으나, 이 경우 미디어 타입을 사전에 설정하여 특정타입만을 선택하도록 해야만 하는 제한점이 있다. 그럼에도 불구하고 사용자가 선택을 할 수 있는 옵션을 더 가질 수 있다는 점은 가치있는 것으로 판단된다.

7. 결 론

본 논문에서는 SMIL의 일부 주요 기능을 객체지향적 분석 설계 방법론[10]에 따라 클래스 모듈로 변환하는 과정을 제시하였다. 먼저 수정 대상이 되는 SMIL 기능을 유형별로 구분하여 새로운 명령형태로 조정하였다. 클립 처리시 동시제어 명령문인 par-access, 순서적 처리 방식인 seq-access, 처리시간을 제한하는 명령문인 time-limit를 추가 하였으며 상대적으로 고속을 요하는 멀티미디어 데이터 처리를 수행할 수 있도록 rate-limit 명령문을 추가시켰다.

기존 SMIL 기능은 많은 장점에도 불구하고 특정 벤더들에 의한 제품화 형태로 개발되었기 때문에 사용자의 요구환경과 잘 맞지 않는 경우에도 제품의 특성을 그대로 수용해야만 하는 제한점이 있다고 할 수 있다. 따라서 사용자가 일부기능을 자신의 환경에 따라 수정하여 사용함으로써 다양성을 넓힌 점이 의미가 있다고 판단된다. 현재 테스트베드 형태로 기능

별로 시험 중인 내용을 타 모듈들로 확대 개발을 한 후 기존 체계와의 비교 검증이 추가로 요구된다.

참 고 문 헌

[1] Jeff Ayars, "Synchronized Multimedia Integration Language (SMIL2.0)", W3C Recommendation, PR-smil20-20010651 07 August 2001.

[2] Luca Cardelli and Rowan Davies, "Service Combinators for Web Computing", IEEE Transactions on Software Engineering, Vol. 25, No. 3, pp.309-316, May-June 1999.

[3] H. Marais, "WebL, A programming language for the Web", pp.259-270, April 1998.

[4] H. Marais - WebL, "A programming language for the Web", Compaq system Research Center, pp.67-68, April 1998.

[5] Real one player, "Real one player guide", URL: http://service.real.com

[6] 방혜자, "멀티미디어 교육을 위한 실시간 영상 강의 시스템의 설계와 구현", 멀티미디어학회 논문지, 제 5권 제 6호, pp.626-630, 2002.

[7] D.C.A. Bulterman, "SMIL 2.0 part2 : Examples and Comparisons", IEEE MultiMedia, Vol.9, No.1, pp.74-84, 2002.

[8] W3C, W3 School, http://www.w3schools.com/default.asp, 2003.

[9] Real one player, "Real one Network Guide", http://service.real.com/help_library/guides/realone/Production_Guide

[10] Ian Sommerville, "Software Engineering", Addison Wesley, 2001.

권 오 현

1975년 해군사관학교 졸업
1980년 미국 NPGS 컴퓨터 시스템 석사
1989년 중앙대학교 전산학과 박사
1978년~2000년 해군 및 국방부 전산분야 근무
2001년~현재 동명정보대학교 컴퓨터공학과 부교수
관심분야 : 객체지향 분석설계, 컴포넌트 시스템 시스템 소프트웨어