제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

기본적인 SCORM 콘텐츠의 모바일 적응화

심진욱*, 장병철*, 이승욱**, 차재혁*
*한양대학교 정보통신 대학원
**한국전자통신연구원(ETRI)
e-mail:guldary@ihanyang.ac.kr

Mobile Adaptation for Basic SCORM Contents

Jinwook Shim*, Byoungchol Chang*, Seung-Wook Lee**,
Jae Hyuk Cha*
*Graduate School of Information and Communications,
Hanyang University
**Electronics and Telecommunications Research Institute

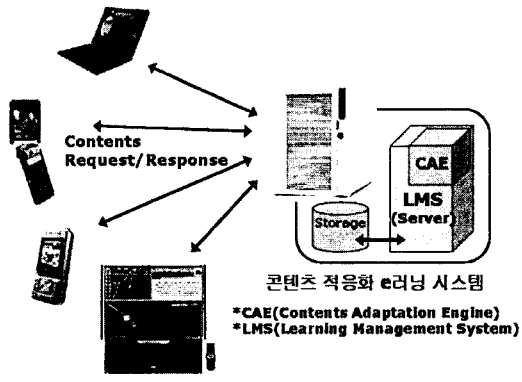
요 약

유비쿼터스 시대의 도래에 발맞춰 유저가 콘텐츠를 이용하는 방법과 사용하는 기기는 PC에 머물지 않고 점차 다양화, 소형화 되고 있다. 최근, 많은 콘텐츠 중에서 e러닝 콘텐츠에 대한 관심은 높아지고 있으며, 본 논문에서는 e러닝 콘텐츠를 다양한 디바이스에 적합한 콘텐츠로 동적 변환하는 모듈을 제안한다. CC/PP 기반의 디바이스 프로파일과 사용자 정보를 포함한 유저 프로파일, 콘텐츠 구성정보를 가진 콘텐츠 프로파일을 통합하여 특성을 추상화 한다. 추상화 된 프로파일을 기반으로 변환방법을 생성 하여 콘텐츠에 적용함으로서 해당 디바이스에 적합한 콘텐츠로 적응화 한다. 기존 HTML의 경우 잘 구성된(Well-formed)문서가 아닌 경우가 많고, 문서 내에 데이터와 구조정보를 모두 포함하고 있으므로, 요구되는 데이터의 추출과 정형화된 변환룰의 적용에 어려움이 따른다. 그 대안으로 본 적응화 모듈은 문서의 데이터와 표현 구조를 분리 할 수 있는 XML/XSL 기반의 콘텐츠를 대상으로 하고 있으며, e러닝 콘텐츠의 특성에 적합한 콘텐츠 프리패치 및 캐시 기법을 적용하여 콘텐츠 동적변환에 따른 응답시간 오버헤드를 최소화 하였다.

1. 서론

네트워크 기술과 모바일기기 성능의 비약적인 발전으로, PC를 통하여 접할 수 있었던 콘텐츠들을 휴대전화, PDA 등을 통하여 손쉽게 접할 수 있게 되었다. 그러나 PC기준 콘텐츠의 직접적인 재사용은 화면사이즈의 한계, 멀티미디어 포맷의 지원 능력의 차이 등 다양한 문제점을 가지고 있다. 이에 따라 디바이스(Device)별로 다양한 환경 프로파일을 수립하여 각각의 사용자 환경에 적합한 콘텐츠로 변환하는 과정이 필요하며, 이에 따른 콘텐츠 적응화 모듈을 제안한다.

본 콘텐츠 적응화 모듈은 변환에 적용되는 대상 콘텐츠를 XML로 작성된 e러닝 콘텐츠로 하였으며, 미국 ADL [1]에서 표준화 하고 있는 SCORM[2] 콘텐츠를 사용하는 LMS(Learning Management System)와 상호 연계되어 있다. 기존의 LMS에 확장 콤포넌트 형태로 탑재되어, 콘텐츠를 요청한 디바이스에 가장 적합한 형태로 변환시키는 역할을 수행한다.



[그림 1] 콘텐츠 적응화 e러닝 시스템

2. 관련연구

2.1· 디바이스 프로파일(Device Profile)

콘텐츠를 다양한 디바이스에 맞게 변환하기 위하여 디바이스 식별 및 기능 인지를 위한 프로파일이 필수적이며, W3C에서 표준화를 진행 중인 CC/PP(Composite Capability/Preference Profiles)[3]가 있다.

2.2 HTML/XML 기반 콘텐츠 변환

콘텐츠를 특정용도로 변환하는 방식으로, HTML 파일의 내/외부에 주석(annotation)파일을 작성하고, 이를 이용하여 변환을 수행 하는 주석 기반의 변환방식(Annotation based Transcoding)[4]이 있다. IBM에서 개발한 주석기반 방식의 변환 시스템으로 WebSphere Transcoding Publisher[5]가 있으며, 이는 RDF/XML 형태의 주석을 작성하고, 해당 주석을 이용하여 다양한 디바이스에 적합한 콘텐츠로 변환한다.

INRIA에서 진행하는 Opera프로젝트에서도 콘텐츠 변환 시스템을 연구하고 있으며, 변환된 최종 출력물을 SMIL[6]형태로 하고, SMIL재생기를 이용하여 멀티미디어 콘텐츠를 표현하는 것이 특징이다[7]. WAM(Web Adaptation and Multimedia)프로젝트로 전환되어 현재 진행 중이다. 유사한 연구로 Cuypers[8]가 있으며 SMIL및 HTML 형태로 변환 가능하다.

3. e러닝 콘텐츠

3.1 SCORM 콘텐츠

SCORM은 ADL에서 표준화 하고 있는 e러닝 콘텐츠의 실질적인 업계표준 이다. 표준 없이 제작된 e러닝 콘텐츠는 재사용이 어렵고, 이종의 LMS에 호환되지 않는다. 따라서 현재 많은 콘텐츠업체들이 e러닝 콘텐츠를 SCORM 규격에 맞추어 제작 하고 있다.

SCORM 콘텐츠는 구조를 정의하고 콘텐츠를 구성하는 세부 자원들을 imsmanifest.xml 파일에 기술하고 있으며, 모든 콘텐츠와 그들 상호간의 관계를 정의하고 있다. imsmanifest.xml 파일의 정보를 바탕으로 콘텐츠에 사용된 세부자원(그림, 플레쉬, 동영상 등)을 분석할 수 있으며, 콘텐츠 변환에 적용 할 모노미디어 판단기준이 된다.

3.2 XML기반 콘텐츠 제작

HTML 문서는 웹페이지의 디자인을 위해 정의된 태그를 가지고 있다. 모델 및 데이터 정보 외에 화면에 어떻게 표현 할 것인가에 대한 스타일 정보를 이미 내포하고 있기 때문에, 스타일정보와 데이터의 혼재로 인한 효과적인 데이터 추출과 해당 데이터를 이용한 변환이 어렵다. 따라서 데이터와 스타일 정보를 분리 할 수 있는 XML과 XSL[9] 구조로 SCORM 콘텐츠 제작을 제안한다. XML 데이터에는 변경을 가하지 않고 XSL을 통한 화면 스타일과 구조정보의 변경으로 다양한 형태의 콘텐츠로 변환이 가능하며, 기 정의된 스키마에 따라 XML 데이터가 구성되어 있기 때문에 해당 스키마 구조에 적합한 정형화된 XSLT 적용이 가능하다.

4. 콘텐츠 적응화 모듈의 설계

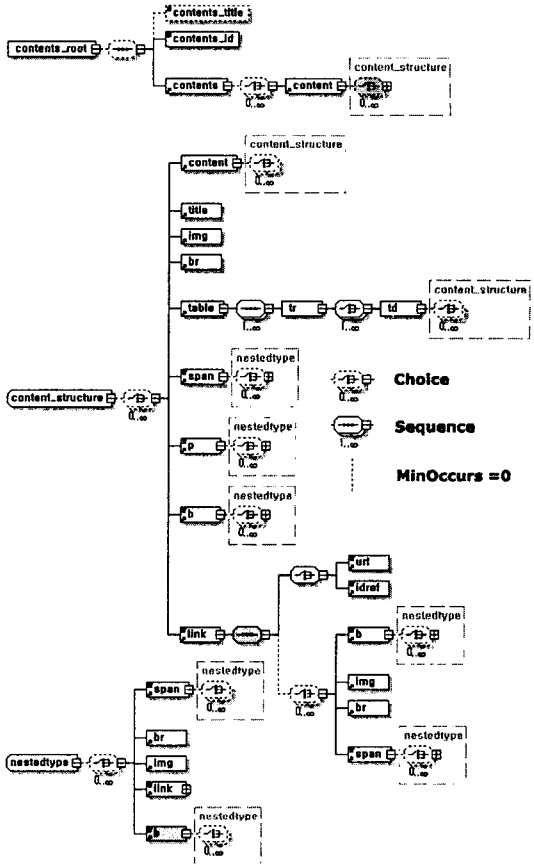
4.1 콘텐츠 스키마

본 콘텐츠 적응화 모듈에 적용하는 콘텐츠 스키마는 XML Schema[10]를 사용하였으며, 그림 2와 같다.

contents_root를 최상위 루트 엘리먼트(Root Element)로 하였고, complex_type 형태로 정의한 content_structure 구조를 사용하여, 반복되는 content 엘리먼트를 정의하였다. 각각의 서브 콘텐츠 내부에서 반복될 수 있는 엘리먼트들은 nestedtype으로 정의하였다.

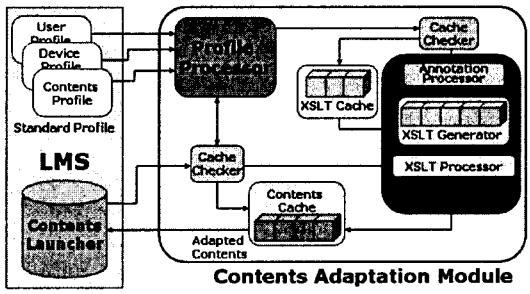
4.2 콘텐츠 적응화 모듈의 전체 구조

본 모듈은 전체적인 e러닝 시스템을 담당하는 LMS에 탑재되는 콤포넌트 형태로 구성된다. LMS는 기본적인 학습관리시스템의 역할을 담당하며, 이종의 디바이스(PC, PDA, DTV, 휴대전화 등)에서의 페이지 요청에 따라 컨



[그림 2] 콘텐츠 스키마 예

텐츠 적응화 모듈에 사용자의 디바이스에 적합한 콘텐츠 변환을 요청한다. 본 모듈은 크게 프로파일 엔진, 변환 엔진, 프리패치 엔진, 캐시 엔진으로 구성되어 있다.



[그림 3] 콘텐츠 적응화 모듈

4.2 프로파일 엔진(Profile Engine)

프로파일 엔진은 클라이언트로부터 얻은 정보를 정의된 스키마에 맞게 매핑하는 프로파일 체커(Profile Checker)와 정의된 프로파일을 구조화하는 프로파일 프로세서(Profile Processor)로 구성된다. CC/PP 프로파일에 기반을 둔 UAProf(User Agent Profile)[11]의 서브셋으로 디바이스 프로파일(Device Profile)을 구성하고, LMS의 스토리지에 저장된 사용자의 학습정보 항목을 참고하여 사용자 프로파일(User Profile)을 생성하며, imsmanifist.xml에

포함된 리소스를 분석하여 콘텐츠 프로파일(*Contents Profile*)을 정의한다.

4.3 변환 엔진(Transformation Engine)

변환엔진은 크게 주석 처리기(Annotation Processor), XSLT 생성기(XSLT Generator), XSLT 처리기(XSLT Processor, Xalan XSLT Processor[12])로 구성되어 있다.

주석 파일에는 Xpath[13]로 세부 콘텐츠 위치를 지정하고 콘텐츠의 변환특성(삭제, 대치 등)을 기술한다. 주석처리기는 작성된 주석 파일을 이용하여, DOM[14] 객체로 매핑한 콘텐츠를 순회하며, 콘텐츠를 변경한다. 변경된 DOM 객체는 XSLT 처리기의 입력 소스로 사용한다.

```
<?xml version="1.0" encoding="EUC-KR"?>
<contents_annotation>
 <contents id="Photoshop_Lesson1">
  <annotation>
   <desc target="/contents_root[1]/contents[1]/content[3]"
         action="delete"/>
   <desc target="/contents_root[1]/contents[1]/content[4]/
         img[1]" action="replace">
     <img src="images/block_file.gif"/>
   </desc>
  </annotation>
 </contents>
 <contents id="Photoshop_Lesson2">
  <annotation>
   <desc target="/contents_root[1]/contents[1]/content[2]"
        action="delete"/>
  </annotation>
 </contents>
</contents_annotation>
```

[그림 4] Annotation 파일 예

기존의 XSL을 이용한 XML문서의 변환 방식은 페이지에 대응되는 XSL문서가 각각 존재하고, 미리 정의된 스키마를 준수하는 경우에 한하여 해당 스키마에 적합한 공통적인 XSL의 적용이 가능하다. 다양한 디바이스에 적용해야 할 경우 디바이스 마다 추가적인 XSL문서 제작이 필요하며, 동일한 디바이스라도 지원특성에 따라 서로 다른 XSL문서가 필요하므로, 변환에 적용 할 XSL문서의 분량은 크게 증가 한다.

본 엔진에서는 페이지마다 미리 정의된 XSL을 사용하는 방식을 사용하지 않고, 세부 XSL Template별로 분류하여 저장하고, 디바이스 특성에 따라 조합하여 동적으로 XSL을 생성하고 적용하는 방식을 사용한다. 세부적인 변경이 있을 때 마다 XSL 문서를 새로 제작하는 대신, 템플릿의 일부만 변경하여 변화된 XSL 문서를 얻을 수 있다.

그림 5는 전체적인 레이아웃을 결정하는 display_page 템플릿을 대체 하는 예이다. A 템플릿은 공통이므로 통합 XSL 생성 시에 필수적으로 포함되며, B-1, B-2는 프로파일을 분석하여, PC환경의 경우 topmenu부분과 leftmenu 부분을 출력하는 B-1 템플릿을 선택하며, 작은 크기의 화면을 가진 PDA와 같은 디바이스의 경우 topmenu와 leftmenu를 생략하고 컨텐츠 내용만 표기하는 너비 240px 레이아웃을 가지는 B-2 템플릿을 선택한다. 추가적인 레이아웃이 필요할 때는 전체적인 XSL을 제작하는 것이 아니라 B-3 템플릿을 추가하고 선택조건에 포함시킨다. A와 B 템플릿은 기본 레이아웃을 선택하는 부분의 예이며, K템플릿은 title 태그를 표현하는 디자인 템플릿의 예다.

템플릿 조각들은 카테고리 별로 분류되어 통합 XSL 템플릿 XML 파일(그림 6)에 저장된다. 통합 XSL 템플릿 파일은 DOM 트리형태로 메모리에 적재하고, DOM API를 통하여 각각의 템플릿 조각들이 존재하는 엘리먼트에 접근 가능하며, 원하는 템플릿 조각데이터를 조합하여 콘텐츠에 적응할 XSL을 생성한다.

주석 처리기 에서 얻은 변경된 콘텐츠 DOM 객체와 XSLT 생성기에서 생성한 XSLT 인스턴스를 XSLT 처리기의 입력소스로 사용하여 콘텐츠를 변환을 수행한다.

A

```
<xsl:template match="contents_root">
 <html>
  <xsl:call-template name="display_header"/>
  <xsl:call-template name="display_page"/>
 </html>
</xsl:template>
```

B-1

```
<xsl:template name="display_page">
 <body>
  <table cellspacing="0" cellpadding="0"  border="0">
   <tr><td><xsl:call-template name="topmenu"/></td></tr>
    <tr><td>
      <table cellspacing="0" cellpadding="0" order="0">
       <tr>
         <td><xsl:call-template name="leftmenu"/></td>
         <td class="maincontent">
          <xsl:call-template name="maincontent"/>
         </td></tr></table>
    </td></tr>
   <tr><td><xsl:call-template name="footer"/></td></tr>
  </table>
 </body>
</xsl:template>
```

B-2

```
<xsl:template name="display_page">
 <body>
  <table cellspacing="0" cellpadding="0" width="240">
    <tr><td class="maincontent">
     <xsl:call-template name="maincontent"/>
    </td></tr>
  </table>
 </body>
</xsl:template>
```

K-1

```
<xsl:template match="title">
 <span class="pageheader">
  <xsl:apply-templates/>
 </span>
</xsl:template>
```

K-2

```
<xsl:template match="title">
 <H1>
  <xsl:apply-templates/>
 </H1>
</xsl:template>
```

[그림 5] XSL Template Fragment 예

```
<?xml version="1.0" encoding="EUC-KR"?>
<xsl_template_set>
 <id>PhotoshopTemplate</id>
 <xsl_subtemplate category="A" count="1">
  <template id="1"><!-A1템플릿내용삽입--></template>
 </xsl_subtemplate>
 <xsl_subtemplate category="B" count="2">
  <template id="1"><!-B1템플릿내용삽입--></template>
  <template id="2"><!-B2템플릿내용삽입--></template>
 </xsl_subtemplate>
          :
</xsl_template_set>
```

[그림 6] Template Fragment 저장 XML File

4.4 프리패치 엔진(Prefetch Engine)

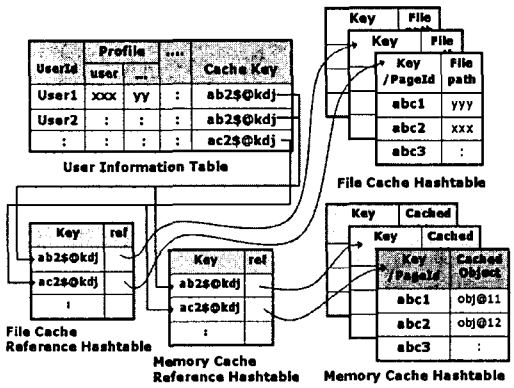
XSLT 및 기타 변환엔진을 이용한 콘텐츠의 동적 변환은 정적 콘텐츠의 처리보다 느리다. 페이지 분기 및 네비게이션 패턴이 다양한 일반적인 웹콘텐츠와 달리, e러닝 콘텐츠는 학습 시퀀스를 가지는 경우가 많으며, 교육적 목적의 콘텐츠이므로 학습을 위하여 페이지마다 지체하는 시간이 타 콘텐츠에 비하여 비교적 길다. 이러한 특징을 활용하여 예상되는 다음 페이지를 미리 변환 처리 하거나, 코스단위로 미리 변환한 뒤에 캐시에 저장 할 수 있다.

4.5 캐시 엔진(Cache Engine)

캐시는 XSLT 인스턴스(instance) 캐시 부분과 콘텐츠 캐시 부분으로 구성되어 있다.

XSLT 인스턴스 생성 비용을 줄이기 위해, 동일한 형태의 XSL을 적용 할 때에는 기존 인스턴스를 재활용 한다.

콘텐츠 캐시는 변환된 결과를 캐시하며, 캐시된 데이터의 저장에 메모리와 파일을 사용한다. 메모리에서 히트 되었을 때 가장 높은 퍼포먼스를 보인다.



[그림 7] 콘텐츠 캐시 구조

사용자에 따른 프로파일을 LMS에서 관리하는 DBMS의 사용자 정보 테이블(User Information Table)에 프로파일 정보와 해당 프로파일을 추상화한 키를 생성하여 함께 저장한다. 하나의 사용자 아이디에 다수의 프로파일을 유지할 수 있으나, 그림 7에서는 단일 프로파일 방식을 기준으로 한다. 사용자 정보 테이블을 이용하여 메모리에 파일 캐시 참조 해시테이블(File Cache Reference Hashtable)과 메모리 캐시 참조 해시테이블(Memory Cache Reference Hashtable)을 생성한다. 사용자별 캐시키를 이용하여 참조 해시테이블을 찾고, 참조 해시테이블에서 메모리상의 데이터 및 파일 경로가 저장된 캐시 해시테이블의 참조객체를 얻는 2단계 참조 방식을 사용한다.

```
Function getCachedContent(user_id, content_id)
begin
1.  key:= getCacheKey(user_id)
2.  mem_ref := getMemoryCacheRef(key)
3.  content := getMemCacheContent(mem_ref, content_id)
4.  if (NULL ≠ content){
5.  }else{
6.      file_ref := getFileCacheRef(key)
7.      file_path := getFileCachePath(file_ref, content_id)
8.      if (NULL ≠ file_path){
9.          content := getContentsFromFile(file_path)
10.     }else{ content := NULL }
11. }
12. return content
end
```

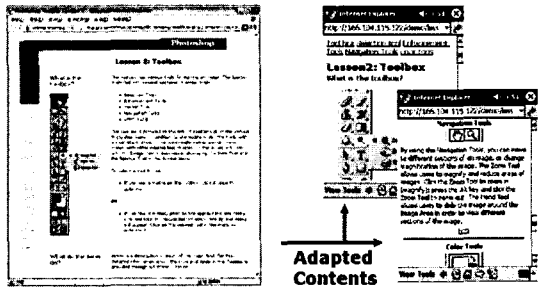
[그림 8] 캐시 콘텐츠 확인 알고리즘

사용자 아이디를 키로 사용하여 사용자 정보 테이블에서 키를 획득(Line [1])한뒤 메모리 캐시를 사용하여 캐시된 콘텐츠를 얻는다(Line [2]-[3]). 메모리 캐시 히트 실패시에는 파일 캐시를 통하여 파일경로를 얻고 해당 파일경로에 위치한 변환된 콘텐츠에 접근한다(Line [6]-[10]).

5. 결론 및 향후연구

본 적응화 모듈에서는 XML 기반 콘텐츠를 다양한 프로파일 정보를 적용하여 동적으로 변환하였다. 변환할 디바이스별로 모든 콘텐츠를 작성할 필요 없이 주석 파일의 적용과 템플릿 조각을 동적으로 조합한 XSL 템플릿을 생성하여 각각의 디바이스에 적합한 콘텐츠를 얻었다.

디바이스, 콘텐츠, 사용자 프로파일의 세 가지 프로파일 정보는 많은 양의 경우의 수를 만들며, 이를 추상화 하여 대표적인 몇몇 형태의 적용타입을 만드는 작업은 큰 어려움이 따른다. 이후 사용자 프로파일과 교육에 적용된 사용자 트래킹 정보를 포함하여 개인화된 콘텐츠 적응화 프레임워크로 확장할 수 있을 것이며. Apache Cocoon[15]과 같은 출판 프레임워크와 결합하여, 내용 측면의 적응화 엔진 역할 뿐만 아니라, HTML로 국한된 출력형태를 PDF, SVG, JPG와 같은 다양한 형태로 표현할 수 있는 출력형태의 다양화 방안도 모색할 수 있을 것이다.



[그림 9] PDA에 적응화 된 콘텐츠의 예

참고문헌

[1] ADL(Advanced Distributed Learning) http://www.adlnet.org/
[2] SCORM(Sharable Content Object Reference Mode) http://www.adlnet.org/index.cfm?fuseaction=scormabt
[3] Composite Capability/Preference Profiles (CC/CP), http://www.w3.org/Mobile/CCPP
[4] Masahiro Hori, Goh Kondoh, Kouichi Ono, Shin-ichi Hirose, Sandeep Singhal "Annotation-Based Web Content Transcoding" 2000
[5] IBM WebSphere Transcoding Publisher http://www-903.ibm.com/kr/software/pvc/wtp/index.html
[6] Synchronized Multimedia Integration Language(SMIL) 2.0, http://www.w3.org/TR/smil20/
[7] Lionel Villard, Cecile Roisin, Nabil Layaida "An XML-based multimedia document processing model for content adaptation" Inria 2001
[8] Joost Geurts, Jacco Van Ossenbruggen, Lynda Hardma "Constraints for Multimedia Presentation" CWI 2002
[9] Extensible Stylesheet Language (XSL) Version 1.0, http://www.w3.org/TR/xsl/
[10] XML Schema, http://www.w3.org/XML/Schema
[11] User Agent Profile - WAPFORUM "UAProf" http://www.openmobilealliance.org/release_program/enabler_releases.html#UAProf
[12] Apache XML Project XSLT Processor "Xalan" http://xml.apache.org/xalan-j/index.html
[13] XML Path Language (XPath) 1.0, http://www.w3.org/TR/xpath/
[14] Document Object Model (DOM), http://www.w3.org/DOM/
[15] XML Web Development Framework "Cocoon" http://cocoon.apache.org/