Journal of Korea Multimedia Society Vol. 9, No. 4, April 2006(pp. 515-527)

데이터 방송 콘텐츠의 호환을 위한 클래스 라이브러리의 설계 및 구현

임현정[†], 임순범[††]

요 약

국내에서 데이터 방송 표준으로 위성 방송은 DVB-MHP를 케이블 방송은 OCAP을 지상파 방송은 ACAP을 채택함에 따라 데이터 콘텐츠의 재사용이 어렵다. 본 논문에서는 이와 같은 문제들을 해결하기 위하여 데이터 방송 콘텐츠 개발을 위한 미들웨어 독립적인 클래스 라이브러리를 설계하고 구현하였다. 애플리케이션과 상이한 미들웨어 사이에서 중계 역할을 할 수 있는 클래스 라이브러리를 정의하기 위해 표준 라이브러리 개발 방법을 활용하고 각 미들웨어 API를 분석하였다. 분석 내용을 바탕으로 내부에 변환 처리 모듈을 포함한 클래스들을 정의함으로써 미들웨어별 라이브러리와 연결이 가능하도록 하였으며 구현한 클래스 라이브러리의 검증을 위하여 테스트 콘텐츠를 구현하여 기존의 콘텐츠와 비교 분석 하였다. 이와 같은 미들웨어 독립적인 클래스 라이브러리를 통한 콘텐츠 개발은 데이터 콘텐츠의 재사용을 용이하게 해주며 각 미들웨어에 대한 개발자의 지식 습득의 부담을 줄여줌으로써 보다 양질의 콘텐츠를 개발할 수 있는 환경 조성이 가능하도록 해준다.

Design and Implementation of A Compatible Class Library for Data Broadcasting Contents

Hyun-Jeong Yim[†], Soon-Bum Lim[††]

ABSTRACT

Domestic standards for data broadcasting differ in the fact that satellite broadcasting chose MHP, cable OCAP, satellite ACAP, therefore making reuse of data contents impossible due to the differences. To resolve this problem, this study designed and implemented independent middleware class library for developing data broadcasting contents. To define class library which could bridge between applications and different middleware, standard library developing methods were used; and analysis of procedural application of each middleware was carried out. Based upon the analysis, classes with embedded transform and handle modules were defined to be linked to libraries of each middleware; and to verify the implemented class library, test contents were created to be compared to the existing contents. This kind of contents development by independent middleware class library allows for reuse of data contents easier, and develops an environment for higher quality of contents by reducing the efforts needed to learn each middleware for the developers.

Key words: Data Contents(데이터방송), Middleware(미들웨어), Class Library(클래스 라이브러리), MHP, OCAP, ACAP

※ 교신저자(Corresponding Author) : 임현정, 주소 : 서울 용산구 청파동2가 숙명여자대학교(140-742), 전화 : 011-9622-9522, FAX : 710-9704, E-mail : hjyim@sookmyung.ac.kr
접수일 : 2005년 9월 5일, 완료일 : 2005년 12월 29일
[†] 준회원, 숙명여자대학교 멀티미디어과학과
[††] 종신회원, 숙명여자대학교 멀티미디어과학과 (E-mail : sblim@sookmyung.ac.kr)
※ 본 연구는 숙명여자대학교 2005년도 교내 연구비 지원에 의해 수행되었음.

1. 서 론

데이터 방송이란 기존의 비디오 콘텐츠와 함께 부가적인 멀티미디어 데이터를 다중화 하여 전송함으로써 시청자들에게 다양한 종류의 디지털 콘텐츠 서비스를 제공하는 방송을 의미한다. 이러한 데이터 방송을 시청하기 위해서 수신자 단말에는 비디오 콘텐츠와 함께 마크업 언어나 자바로 구성되어 있는 데이터 콘텐츠를 적절하게 디코딩하고 표현해 주는 모듈인 미들웨어(Middleware)가 필요하다. 국내 데이터 방송의 경우 서비스 매체에 따라 위성 방송은 DVB-MHP를 케이블 방송은 OCAP을 지상파 방송은 ACAP을 표준으로 채택하고 있다. 그러나 이처럼 상이한 미들웨어 규격으로 인하여 몇 가지 문제점들이 발생하게 된다.

첫째, 별도의 코딩 과정을 필요로 하여 콘텐츠 개발을 위한 고비용 구조를 형성한다. 따라서 동일한 콘텐츠를 두 개 이상의 방송환경으로 콘텐츠를 송출하고자 할 때에는 매번 각 표준에 맞도록 별도의 코딩 과정을 거쳐야하며, 호환 프로그램을 통해 다른 표준으로 변환 시에는 매체에 적합한 형태로 코드를 수정하는 변환 비용이 추가된다.

둘째, 데이터 콘텐츠의 재사용성이 떨어진다. 디지털 방송에 사용되는 비디오 콘텐츠는 MPEG-2를 공통적으로 사용하여 재사용이 가능한 반면 데이터 콘텐츠는 각 표준마다 상이한 규격을 요구하므로 호환성이 떨어진다.

셋째, 양질의 콘텐츠 개발이 어려워 데이터 방송 콘텐츠의 부족 현상이 야기된다. 콘텐츠 개발자들은 간단하지 않은 위성, 지상파, 케이블 방송의 미들웨어 표준들의 상이한 내용을 숙지하고 콘텐츠 개발을 해야 한다는 부담감을 갖고 익숙하지 않은 데이터 방송 콘텐츠 제작 환경에 적응해야 한다.

이러한 문제점이 발생하지 않도록 모든 매체에서 공통적으로 적용되는 데이터 방송 표준 규격의 사용이 가장 바람직했을 것이나, 각 매체별로 이미 상이한 전송 및 SI(Service Information) 규격에 기반하고 있는 상황에서 국내 적용을 위한 제 3의 데이터 방송 표준을 만드는 것이 바람직하지 않다는 현실과 데이터 방송의 조속한 활성화 및 기술 개발 방향의 제시를 목적으로 표준을 결정하게 된 것이다.

위와 같은 문제점을 해결하기 위하여 본 논문에서는 미들웨어 독립적인 클래스 라이브러리 통해 기존의 방송 규격을 수용하여 호환성을 유지할 수 있는 기술들을 확보함과 동시에 보다 다양한 콘텐츠를 효율적으로 제작할 수 있는 방법을 소개하고자 한다. 이를 위해 2장에서는 각 미들웨어 특징을 살펴보고, 3장에서는 각 미들웨어의 절차적 응용에 대하여 비교 분석하였다. 4장에서는 클래스 라이브러리의 설계와 구현에 대하여 소개하고, 설계한 클래스 라이브러리 검증을 위한 테스트용 콘텐츠를 구현하여 다중 플랫폼에서의 테스트 및 기존의 콘텐츠와 비교 분석하였다.

2. 각 미들웨어의 특징 및 관련 연구 소개

본 연구에서 설계하고자 하는 클래스 라이브러리 정의를 위한 기반 연구로써 먼저 여러 표준화 단체에서 정의하여 현재 데이터 콘텐츠 개발에 이용되는 대표적인 미들웨어인 MHP[1], OCAP[2], ACAP[3]의 간략한 소개와 특징에 대해서 알아본다.

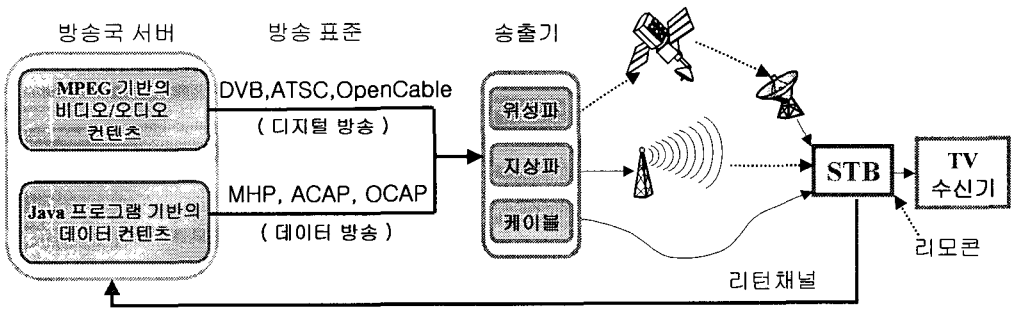


그림 1. 데이터 방송 개념도

2.1 DVB-MHP (Digital Video Broadcasting - Multimedia Home Platform)

DVB는 ELG(European Launching Group)에 의해 디지털 방송의 공개 공동 기술 개발을 위해 설립된 조직 명칭이다. 1997년 DVB는 홈쇼핑, EPG (Electronic Program Guide) 등과 같은 인터랙티브 애플리케이션을 모든 종류의 단말기에서 동작될 수 있도록 제작하고 방송할 수 있는 공통의 API를 정의하기 시작하였다. 1997년 DVB CM(Commercial Module)내에 DVB-MHP[1] 특별 위원회가 결성되면서 기술적인 부분은 DVB TM(Technical Module) 내에 TAM(Technical Aspects of MHP)이 만들어져 DVB API 규격화 작업을 수행하게 되었다. DVB의 데이터 방송인 DVB-MHP에서는 증보된 방송, 대화형 방송, 인터넷 접속에 중점을 두어 표준을 규정하고 있으며 2000년 7월에 MHP 1.0 규격을, 2001년 10월에 MHP 1.1 규격을 발표했다.

DVB-MHP는 단말기의 특성 및 애플리케이션의 복잡도에 따라 리턴 채널을 사용하지 않고 단말기 내에서의 상호작용만을 지원하는 Enhanced Broadcast Profile(MHP 1.0), 물리적 네트워크에 무관하게 리턴 채널을 사용하는 Interactive TV Profile(MHP 1.0), 데이터 방송 서비스와 더불어 인터넷을 통한 데이터 서비스를 제공하는 Internet Access Profile(MHP 1.1)로 나뉜다.

2.2 OCAP (OpenCable Application Platform)

미국은 디지털 케이블 방송의 개방형 표준설립을 위하여 CableLabs에서 1997년 9월 OpenCable을 조직하여 표준화 작업을 시작하였으며 2002년 1월 OpenCable 및 OCAP 1.0 표준방식 최종 발표하였다. OCAP은 케이블 TV 방송에서 양방향 서비스를 위한 애플리케이션 제작의 기반이 되는 표준으로써 유럽의 DVB-MHP 미들웨어 표준을 근간으로 규격을 작성하였으며 현재는 OCAP 2.0을 작업 중이다.

OCAP 1.0[2]은 DVB-MHP 1.0에 기반하고 있으며 OCAP 2.0은 OCAP 1.0 및 DVB-MHP 1.1에 기반하고 있다. OCAP 2.0의 특징은 OCAP 1.0이 정의하지 않은 HTML을 DVB-HTML에서 채용하여 정의하고 있다는 것이다. OCAP은 AIT Signaling, Application database 등은 DVB-MHP 1.0을 대부분 수용했으며 Unbound applications, AIT-like signaling for attributes, ATVEF signaling for attributes and control, Monitor application 등을 위한 자체 API를 정의하였다.

2.3 ACAP (Advanced Common Application Platform)

ACAP[3]은 지상파와 케이블에서 데이터 방송 서비스를 위한 공통 플랫폼을 제공하는 미들웨어 표준으로써 2002년 11월 26일 ATSC와 Cablelab간의 합의에 의해 DASE/OCAP 간의 공통 규격을 만들고자 추진되었다. 원래 명칭은 DCAP(DASE+OCAP)이었으나 표준화 최종 단계에서 케이블TV 방송 진영의 요구를 수용하여 OCAP의 발전된 규격이란 뜻을 가진 ACAP이라 명명되었다.

기술적으로는 GEM과 DASE를 기반으로 OCAP의 일부 기능을 추가한 형태로 표준 규격을 정의하였으며 자바 애플리케이션 형태의 절차적 응용인 ACAP-J와 XHTML 형태의 선언적 응용인 ACAP-X, 그리고 다른 표준과 다르게 혼합 응용을 추가하여 ACAP-X와 ACAP-J의 통합 형태의 애플리케이션

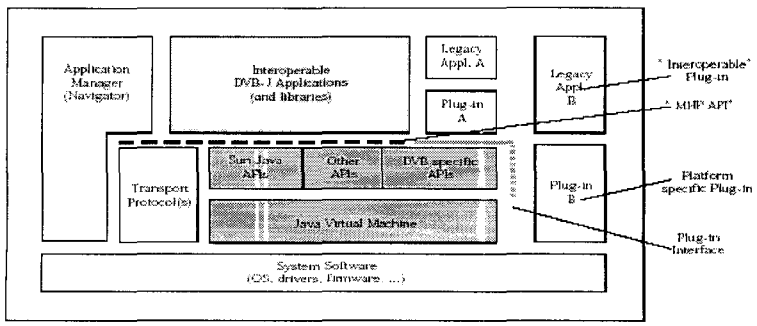


그림 2. DVB-J Architecture

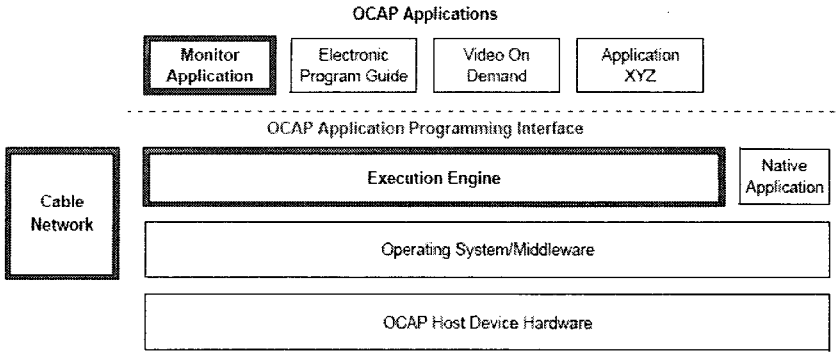


그림 3. OCAP Architecture

에 대해서도 정의하고 있다. 국내에서는 이미 EBS가 2004년 6월 'EBS 장학퀴즈'를 시작으로 ACAP 방식에 의한 데이터 서비스를 시작한 상태이며, 2005년 8월에 표준화가 완료되어 국내에서도 다양한 데이터 방송 서비스의 조속한 활성화를 위하여 ACAP을 지상파 표준으로 채택하였다.

3. 미들웨어 API의 비교 분석

데이터 방송 미들웨어에서 사용되는 콘텐츠 형식에는 마크업 언어 형태의 선언적 응용인 DA (Declarative Application)와 자바 클래스 파일 형태의 절차적 응용인 PA(Procedural Application)가 있다. 대부분의 표준에서 선언적 응용은 선택 사항이기 때문에 본 연구에서는 절차적 응용만을 연구 범위로 정하였으며 MHP 1.0.3[1]과 MHP 1.0 test suit를 그대로 수용하고 있는 OCAP 1.0[2]과 ACAP[3]을 비교 분석 하였다. 각 미들웨어를 살펴 본 결과 다음과 같이 크게 모든 표준에서 공통적으로 사용되는 API와 각 미들웨어별로 추가 정의한 API가 존재하였으며 세부 분석 내용은 다음과 같다.

3.1 공통으로 사용되는 API

MHP, OCAP, ACAP은 모두 DVB-GEM을 기반으로 정의된 최종 사양이므로 GEM에 해당하는 부분을 공통분모로 하고 있다. DVB-GEM은 상이한 기관에서 정의한 GEM 기반의 표준들 사이에 상호 운용성을 최대화하기 위한 목적으로 Basic architecture, Transport protocol, Contents format, Application model 등 다양한 내용을 정의하고 있다.

또한 각 표준에서는 DAVIC[6], HAVi[7], JavaTV [8], JMF[9] 패키지 등을 공통으로 사용하고 있으며 세부 API 내용은 표 1과 같다. 이와 같은 API들의 기능을 분석해보면 먼저 DAVIC은 자원 할당 제어 및 TV A/V 고유의 제어 기능을 추가하여 접근 제어에 관한 클래스를 정의하고 있고, HAVi는 TV 고유의 특성을 반영한 UI 및 화면 모델 제어, 텍스트 레이아웃, 리모콘 입력 이벤트 표현에 관한 클래스 정의하고 있다. JavaTV는 SI 정보 접근, 서비스 선택, 응용 프로그램 주기 제어에 주로 사용되는 클래스 정의하고 있으며, JMF는 TV의 A/V를 제어하기 위한 클래

표 1. 공통으로 사용되는 API

	APIs
DAVICpart09	org.davic.media org.davic.mpeg org.davic.mpeg.sections org.davic.net org.davic.net.tuning org.davic.resources
HAViLevel2	org.havi.ui org.havi.ui.event
JavaTV1.0	javax.tv.carousel javax.tv.graphics javax.tv.locator javax.tv.media javax.tv.media.protocol javax.tv.net javax.tv.service javax.tv.service.guide javax.tv.service.navigation javax.tv.service.selection javax.tv.service.transport javax.tv.util javax.tv.xlet
JMF	javax.media javax.media.protocol

스들로 구성되어 JavaTV API의 기반이 되고 있다.

3.2 미들웨어 종속적인 API

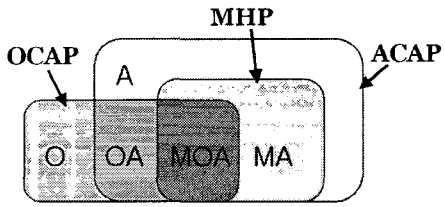
MHP, OCAP, ACAP은 모두 DVB-GEM을 기반으로 정의된 최종 사양이므로 GEM에 해당하는 부분을 공통분모로 하고 있으나 이중 일부는 각 미들웨어에 적합한 형태로 확장하고 재정의 되었으며 필요에 따라 GEM이 포함하고 있지 않는 부분을 새로 정의하였다.

먼저 MHP는 GEM을 바탕으로 유럽 방송 환경에 적합한 SI(Service Information), CA(Conditional Access, 수신제한모듈), 네트워크 프로토콜을 위한 내용 및 사용자 선호도, 보안, 입출력장치 제어, 데이터 접근 제어 등의 기능을 위한 API를 추가 정의하였다. OCAP은 케이블 방송 환경의 특성을 반영하기 위하여 GEM 및 MHP의 API를 기반으로 필요한 API를 추가 정의하였으며, 추가 정의된 내용으로는 OpenCable SI, POD, OpenCable 네트워크 프로토콜, Unbound application, Monitor application 등이 있다. 다음으로 ACAP은 MHP에 정의된 API들을 지원함과 동시에 OCAP에서 케이블 방송 환경을 위한 API를 지원하기 위하여 ACAP-C(Cable)에서 OCAP의 API인 Closed Captioning, Locators, Events, Content Identification API, Extended SI API를 확장하여 지원하고 있다.

이처럼 포함관계를 형성하는 각 미들웨어간의 상이한 API를 더욱 효율적으로 분류하기 위하여 본 논문에서는 그림 4와 같은 영역 기호를 사용하였으며 이에 따라 정리한 미들웨어 종속적인 API의 상세 내용은 표 2와 같다.

4. 클래스 라이브러리의 설계 및 구현

4장에서는 앞장에서 조사한 미들웨어들의 API 관계 분석 자료와 개발자의 편의성을 고려하여 클래스들을 설계하였다. 또한 정의된 클래스 라이브러리를 바탕으로 데이터 콘텐츠를 제작하여 라이브러리 검증을 위한 테스트 콘텐츠를 구현하였으며, 성능 평가를 위하여 기존의 API를 사용하여 개발한 동일한 내용의 콘텐츠와 비교 분석하고 다중 플랫폼 지원 여부



※ 영역 기호에 대한 설명
- MOA: MHP, OCAP, ACAP에서 사용하는 MHP API
- MA: MHP, ACAP에서 사용하는 MHP API
- OA: OCAP, ACAP에서 사용하는 OCAP API
- O: OCAP에서만 사용하는 API
- A: ACAP에서만 사용하는 API
  ACAP-X를 지원하기 위한 API들로 연구 범위에 벗어남

그림 4. 미들웨어 종속적인 API의 상관관계

표 2. 미들웨어 종속적인 API의 분류

분류	APIs	분류	APIs
MOA	org.dvb.lang org.dvb.application org.dvb.net org.dvb.media org.dvb.event org.dvb.dsmcc org.dvb.io.persistent org.dvb.net.rc org.dvb.user org.dvb.io.ixc org.dvb.ui org.dvb.test org.dvb.net.tuning	MA	org.dvb.si org.dvb.net.ca
		OA	org.ocap.si org.ocap.net org.ocap.media org.ocap.ui.event
		O	org.ocap org.ocap.application org.ocap.event org.ocap.hardware org.ocap.hardware.pod org.ocap.service org.ocap.resource org.ocap.system

를 검증하였다.

4.1 클래스 라이브러리 설계

미들웨어 독립적인 API를 설계하기 위하여 구현된 API는 OpenGL이나 Java3D와 같은 표준 라이브러리 개발 방법을 활용하여 애플리케이션과 미들웨어 사이에 다리 역할을 하도록 Bridge pattern을 이용하여 설계하였다. 3D 그래픽 라이브러리인 Java3D는 OpenGL이나 DirectX를 기반으로 하여 자바로 구성된 상위 API를 제공해줌으로써 OpenGL이나 DirectX에 대한 기반 지식 없이도 Java3D 라이브러리를 통해 3D 콘텐츠 구현이 쉽도록 해준다. 본 연구에서도 이와 같은 표준 라이브러리 개발 방법을 활용함으로써 미들웨어 규격에 따른 콘텐츠 제작을 해야하는 번거로움을 없애주고 미들웨어 규격에 대한 지식 습득의 부담을 줄일 수 있도록 하였다.

또한 개발자가 직접 생성해야 할 객체와 시스템 하부적인 백그라운드 객체의 분리를 통해 미들웨어 전반에 걸친 디지털 방송 시스템 환경적인 내용 설정을 캡슐화하여 은닉함으로써 개발자에게 관리의 용이성 및 편리한 접근성을 제공하도록 하였다. 설계한 라이브러리를 통해 콘텐츠 개발 시 개발의 효율성을 높일 수 있도록 실제 데이터 콘텐츠 제작 업체에서 필요한 기능 및 요구 사항을 분석하여 설계에 반영하였다.

4.2 API 및 세부 클래스 정의

MHP, OCAP, ACAP이 GEM을 공통 분모로 하고 있으므로 이를 바탕으로 각 방송 환경의 여러 가지 상황을 고려하여 클래스 라이브러리를 설계하는 것이 타당할 것으로 보인다. 또한 클래스 라이브러리의 설계를 위하여 자바 애플리케이션 작성 경험은 있으나 데이터 방송 환경이 낯선 개발자가 콘텐츠 개발 시 사용하기 편리한 기능을 제공하여 효율적으로 작업할 수 있도록 기능 위주로 클래스들을 설계하였다. 또한 실제 개발 필드에서 필요로 하는 기능 및 요구 사항을 조사하여 조사 결과를 근거로 바탕으로 환경 설정 프리셋팅, bufferedImage 생성 풀링 기능, 연속키/복합키 기능 등의 기능을 제공하는 클래스들을 정의하였다. 이렇게 정의한 클래스들은 기능에 따라 9개의 패키지로 그룹 지었으며 본 논문에서는 패키지 명칭을 GEM, MHP, OCAP, ACAP에서 API 분류 시 사용하는 카테고리 이름을 참고하여 패키지 이름을 정하였다. 각 API의 특징과 세부 클래스들을 간단히 소개하면 다음과 같다.

(1) Graphic User Interface (GUI) API

Background Layer, Video Layer, Graphics Layer를 위한 그래픽 환경 설정 초기화와 이미지 및 그래픽스를 위한 클래스들을 정의하였다.

- GraphicEnvironment : 그래픽 환경 초기화 설정 및 백그라운드 디바이스와 그래픽 디바이스 설정을 위한 클래스이다.
- PopupWindow : Graphics Layer 위에 웹 브라우저의 팝업창과 같은 기능을 제공 한다.

(2) Streamed Media API

스트림 미디어를 실시간으로 제어하고 이벤트 관리를 위한 클래스들로 구성되어있다.

- MediaController : 스트림 미디어의 재생, 정지, 반복, 부분 재생 등의 기능을 제공한다.
- MediaManager : 스트림 미디어를 사용하는 모든 클래스를 생성 및 관리한다.

(3) Sound API

스트림 미디어 형태가 아닌 애플리케이션에서 파일 형태로 접근하는 사운드의 환경 설정, 재생, 정지 등 사운드 컨트롤을 위한 클래스들을 포함한다.

- Sound : Hsound를 내부적으로 생성하여 소리의 재생, 정지, 자원 해지 등을 쉽게 제어할 수 있는 기능 제공한다.

(4) Contents Resource API

사운드,.이미지 등 멀티미디어 데이터에 대한 접근과 애플리케이션 실행 후 리소스 반환을 위한 클래스들을 포함한다.

- ResourceManager : 모든 클래스에서 사용하는 자원을 생성 및 해지를 위한 클래스이다.
- ImagePool : 이미지의 HashTable ID 값과 이미지 보관을 위한 클래스로 계속 사용되는 이미지와 한번 사용 후 자원 해지 되는 이미지에 대한 정보 제공한다.

(5) Process Control API

애플리케이션의 라이프 사이클 구현과 콘텐츠 서비스 분야에 따른 핵심 알고리즘을 위한 클래스들을 포함한다.

- Xlet : Xlet 인터페이스를 구현한 클래스로 실

제 하나의 애플리케이션 라이프 사이클을 갖는 클래스로 메인 쓰레드를 포함하고 있다.

• Core : 콘텐츠의 특성에 맞는 애플리케이션의 핵심 알고리즘이 구현되는 클래스이다.

(6) Key Event API

연속키, 키 초기화 및 입력, 특수키 맵핑, 디스플레이의 동기화를 위한 클래스들을 정의하였다.

• ChainKey : 리모콘의 버튼을 계속 누르고 있을 때 이를 연속적인 이벤트로 받아들여 처리하는 기능을 제공한다.

• WaitBreak : 애플리케이션이 종료 상태 및 무한 반복, 일시 정지 상태일 때 버튼 이벤트를 통해 Pause 상태를 빠져 나오는 기능을 제공한다.

(7) Data Access API

모든 파일 입출력과 시스템간의 커뮤니케이션을 위한 클래스들을 포함한다.

• DataInputStream : 데이터 통신을 위해 캡슐화된 메시지를 각각의 프로토콜 정의에 맞게 데이터를 패킷 단위로 분리 저장하는 기능을 제공한다.

• DataManager : 데이터 처리와 관련된 기능을 수행하는 모든 클래스들의 제어와 메시지 반환에 대한 관리 기능을 제공한다.

(8) Service information and selection API

다중화된 전송 스트림에서 특정 서비스를 선택하고 SI table에 대한 접근 등의 기능을 제공하는 클래스들을 정의하였다.

• SIManager : 로케이터 값을 통한 SI 테이블 정보 반환과 서비스 이동을 위한 메소드 등을 제공한다.

• SITable : SI 정보를 애플리케이션에 비동기적으로 제공하고 SI 정보가 변경될 경우 애플리케이션에게 통보하는 매커니즘을 제공한다.

(9) Return Channel API

사용자와의 인터랙티브한 콘텐츠 구현을 위해 필요한 클래스들을 정의하였다.

• Message : 리턴 채널을 통해 전달되는 메시지가 외부에 노출되지 않도록 캡슐화 하는 인터페이스이다.

• Network : 메시지를 서버에 전송하기 위한 여러 통신 환경 설정을 위한 클래스이다.

4.3 클래스 라이브러리 API의 내부구현

3장에서 분석한 바와 같이 각 미들웨어에서 사용하는 패키지는 공통으로 사용되는 집합이 존재하고 상이한 패키지 집합이 존재한다. 따라서 구현될 클래스 역시 크게 공통으로 사용하는 패키지를 참조하여 구현 가능한 클래스와 미들웨어 종속적인 패키지를 참조하여 구현되는 클래스로 구분할 수 있다.

먼저공통으로 사용하는 패키지를 참조하여 구현되는 경우는 표 2의 MOA에 속하는 클래스들을 상속 확장하여 구현되는 클래스로 콘텐츠 개발 시 요구사항으로 조사되었던 클래스 단일화 기능을 제공한다. 여기서 클래스 단일화 기능이란 개발에 필요한 기능 위주로 여러 클래스를 하나의 클래스로 감싸 놓은 형태인 상위 클래스들을 제공하는 것으로서 이를 통해 개발이 더욱 쉽고 효율적으로 이뤄질 수 있다. 그 대표적인 예가 그림 5의 PopupWindow 클래스이며 여기서는 공통적으로 사용되는 dvd 패키지, havi 패키지 등을 임포트하여, 팝업창의 생성 및 내용 구성

```
1  //PopupWindow.java
2
3  import java.awt.Color;
4  import java.awt.FontMetrics;
5  import java.awt.image;
6  import java.awt.image.ImageProducer;
7  import org.dvb.ui.DVBAlphaComposite;
8  import org.dvb.ui.DVBBufferedImage;
9  import org.dvb.ui.DVBGraphics;
10 import org.havi.ui.Hcontainer ;
11 import org.havi.ui.HSound ;
12 import org.havi.ui.event.HKeyListener;
13 import itv.gui.Button;
14 import itv.gui.Animation;
15 import itv.gui.XYLayout;
16
17 public class PopupWindow extends Hcontainer implements HkeyListener
18 {
```

그림 5. 공통으로 사용하는 패키지를 참조하여 구현된 클래스

이 용이하도록 클래스를 정의하였다. 이와 같은 클래스 단일화 기능을 이용하여 실제 애플리케이션 개발에 유용한 기능 위주로 클래스를 제공함으로써 보다 상위 레벨에서 효율적인 콘텐츠 개발이 가능하다.

다음으로 미들웨어 종속적인 패키지를 참조하여 구현되는 경우에는 각 미들웨어에 API의 기능을 비교 분석하여 제공하는 기능이 유사한 경우와 기능이 상이한 경우에 따라 클래스 구현 방법에 차이가 있다. 기능이 유사할 경우는 표 2의 MA나 OA를 참조하는 경우에 해당하는 것으로써 미들웨어에 따라 다른 메소드를 호출하거나 다른 반환 값을 전달해주는 변환 처리 모듈을 구현해야 한다. 예를 들어 MHP의 SI패키지와 OCAP의 SI패키지는 서로 다르지만 둘 다 Service Information을 제공해주기 위한 패키지라는 공통점이 있기 때문에 기능이 유사한 패키지로 분류된다.

기능이 상이할 경우에는 해당하는 클래스들을 bypass 형태의 연결이 필요했으며 표 2의 O를 참조하는 클래스 구현의 경우에 해당한다.

4.4 테스트용 콘텐츠 구현 및 테스트

설계한 클래스 라이브러리의 검증을 위하여 먼저 다중 플랫폼에서의 실행 여부를 테스트하여 클래스 라이브러리의 호환성을 입증하고, 기존 개발 환경에서 개발된 콘텐츠와 비교분석하여 개발의 효율성을 평가하였다.

4.4.1 다중 플랫폼 지원 여부 테스트

개발된 표준 라이브러리를 이용하여 개발한 콘텐츠를 MHP 플랫폼과 OCAP 플랫폼에서 각각 실행하여 다중플랫폼을 지원하는 표준 라이브러리의 기능을 검증하였다. 그림 8은 디지털 방송 콘텐츠 개발을 위해 갖추어야 할 개발 환경으로 Stream Station은 AV(Audio Video) 데이터와 콘텐츠를 동시에 송출하며 Application Server는 콘텐츠에 필요한 데이터를 공급하는 역할을 한다.

설계한 클래스 라이브러리의 검증을 위해 오목 게임 콘텐츠를 구현하여 소스코드를 컴파일하고, 컴파일된 콘텐츠를 OCAP 용 스트리밍 장비와 MHP용 스트리밍 장비를 이용하여 송출하였다. 송출한 결과 동일한 콘텐츠가 서로 다른 플랫폼에서 실행됨으로써 표준 라이브러리의 호환성을 검증하였다.

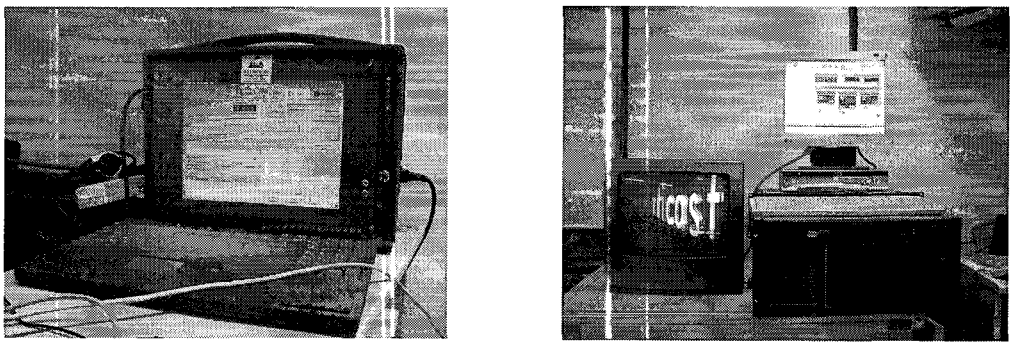
또한 본 연구에서는 현재 상용화 되어있는 ACAP 셋탑박스가 없는 관계로 ACAP 환경에서의 실제 컨텐츠 송출 및 테스트가 어렵기 때문에 클래스 라이브러리의 검증을 위하여 구현한 콘텐츠를 PC용 애플리

```
public Object getSMTable(String property){
    if(currentSi == DVB_SI){
        Locator locator = null;
        SITable table = null;
        try {
            locator = LocatorFactory.getInstance().createLocator((java.lang.Syste
            try {
                table = (SITable) SIManager.createInstance().getService(locator);
            } catch (SecurityException e) {
                e.printStackTrace();
            } catch (InvalidLocatorException e) {
                e.printStackTrace();
            }
        } catch ( MalformedLocatorException ex ) {
            ex.printStackTrace();
        }
        return table;
    }else{
        DvbLocator locator = null;
        SITable table = null;

        try {
            locator = new DvbLocator(property);
            if(locator != null){
                System.out.println("<<DVBLocator>> : Network Id - "+locator.getOr
                System.out.println("<<DVBLocator>> : File Path - "+locator.getFil
                System.out.println("<<DVBLocator>> : File Path - "+locator.getFil
            }
        } catch (org.davic.net.InvalidLocatorException e1) {
            e1.printStackTrace();
        }
```

그림 6. SI 변환 모듈

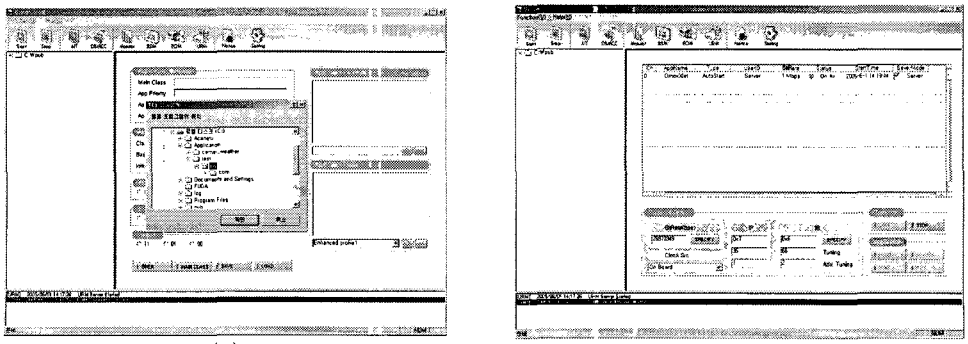
```
//hardware.java

import org.ocap.hardware.CopyControl;
import org.ocap.hardware.Host;
import org.ocap.hardware.PowerModeChangeListener;
import org.ocap.hardware.VideoOutputPort;
import org.ocap.hardware.pod.POD;

public class Hardware {
    private static Hardware hard = new Hardware();
    public static Hardware getInstance(){
        return hard;
    }

    public String checkSISystem(){
        return null;
        }
    public CopyControl getCopyControl(){
        return null;
        }
    public Host getHost(){
        return null;
        }
    public PowerModeChangeListener getPowerModeChangeListener(){
        return null;
        }
    public VideoOutputPort getVideoOutputPort(){
        return null;
        }
    public POD getPOD(){
        return POD.getInstance();
        }
}
```

그림 7. bypass 형태의 연결



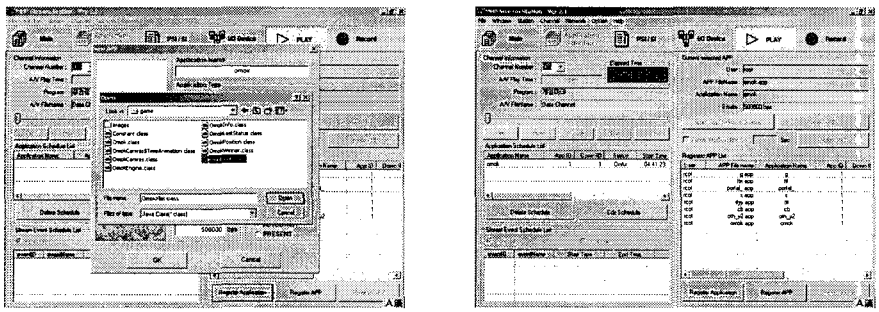
(a) (b)

그림 8. 실험 환경: (a) MHP 개발 환경, (b) OCAP 개발 환경



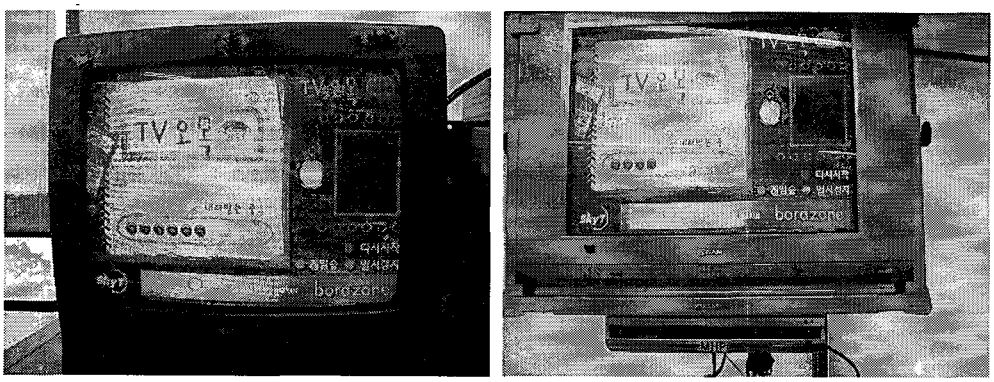
(a) (b)

그림 9. OCAP 장비에서 (a) 콘텐츠 등록 및 (b) 송출 화면



(a) (b)

그림 10. MHP 장비에서 (a) 콘텐츠 등록 및 (b) 송출 화면



(a) (b)

그림 11. 콘텐츠 실행 화면:. (a) OCAP 셋탑박스에서 실행 화면, (b) MHP 셋탑박스에서 실행 화면

케이션 상태에서 컴파일 에러가 발생하지 않도록 하였다. 따라서 추후에 ACAP 셋탑박스가 상용화 된 후에 본 논문에서 구현한 클래스 라이브러리의 ACAP 시스템에서의 적용이 어렵지 않을 것이라 예상한다.

4.4.2 기존 콘텐츠와의 비교

기존 개발 환경에서 개발된 콘텐츠와 제안된 클래스 라이브러리를 통해 개발된 콘텐츠를 비교하여 기존의 콘텐츠와 그래픽, 인터랙션, 로딩시간 등에 있어서 어떤 차이가 있는지, 제안한 클래스 라이브러리를 통해서 콘텐츠 개발 시에 어느 정도 개발의 효율성을 가져다주는지에 중점을 두어 다음과 같은 항목을 비교 분석 하였다.

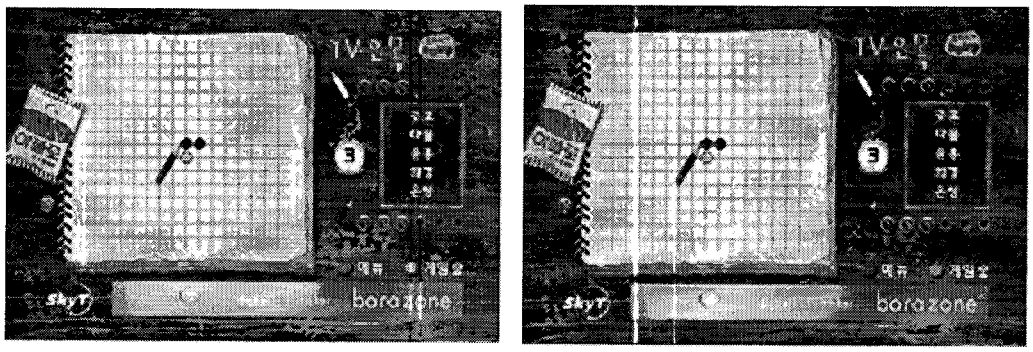
(1)그래픽디스플레이및인터랙션기능

테스트 결과 그래픽 디스플레이 및 제공하는 기능에 있어서 두 콘텐츠간의 차이가 없었다. 일반적으로 게임 콘텐츠는 다른 콘텐츠들에 비해 다양한 인터랙션과 화려한 그래픽을 제공한다. 따라서 뉴스, 증권 등 다른 서비스 도메인에서 방송될 콘텐츠 제작 시에도 본 논문에서 설계한 클래스 라이브러리를 사용하여 원하는 그래픽과 인터랙션 기능 등을 무리 없이 표현할 수 있을 것이라 예상한다.

(2)사용된클래스개수

다음으로 개발의 편의성을 비교해보기 위하여 개발자가 고려해야할 클래스 개수와 실제 개발 시 임포트(import)된 클래스 개수를 비교하였다. 표 3이 나타내는 것처럼 국내 데이터 방송 환경에서는 서로 다른 미들웨어의 상이한 API에 대한 이해가 필요하므로 개발자는 총 587개의 클래스를 고려하여 콘텐츠 개발을 해야 한다. 그러나 본 논문에서 제안한 클래스 라이브러리는 총 47개의 클래스로 구성되어있다.

오목 게임 콘텐츠 구현 시 임포트 되는 클래스는 기존의 MHP 사용 시에는 42개의 클래스를 임포트



(a) (b)

그림 12. 그래픽 디스플레이 비교: (a) 기존의 오목 게임 콘텐츠, (b) 새로 구현된 오목 게임 콘텐츠

표 3. 데이터 콘텐츠 개발자가 고려해야했던 클래스 개수

API	DAVIC	HAVi	JavaTV	JMF	MO	MA	OA	O	합계
클래스 개수	87	107	102	66	129	38	24	34	587

해야 했으며 본 논문에서 제안한 라이브러리를 통해 구현 시에는 15개의 클래스를 임포트 한다. 이처럼 많은 수의 클래스를 줄일 수 있었던 이유는 먼저 변환처리 모듈을 통해 개발자가 상이한 미들웨어 환경에서 알아야했던 API들에 대한 이해가 필요 없었기 때문이며, 다양한 데이터 콘텐츠를 개발해온 경험을 바탕으로 데이터 콘텐츠 개발에 실제 자주 사용되는 클래스들과 그렇지 않은 클래스들을 파악하고 개발 필드에서 필요로 하는 기능을 위주로 클래스를 단일화함으로써 하나의 클래스 접근으로 원하는 기능을 구현할 수 있도록 설계하였기 때문이다. 또한 디지털 방송 환경 설정에 관해 임포트 해야 했던 많은 클래스들을 캡슐화하여 은닉했기 때문이다.

(3)애플리케이션크기및코딩라인

현재 스카이라이프는 디지털 방송 서비스를 위해 총 12기의 위성 중계기를 임대해주고 있고, 이 가운데 데이터 방송은 1개의 중계기를 할당받아 사용 중이다. 1기의 중계기는 대략 38 Mbps의 대역폭을 가지고 있으며 많은 업체가 중계기를 함께 사용하는 이유로 콘텐츠는 보통 400~500KB로 구성되는 것이 일반적이다.

전체 애플리케이션의 크기를 측정하는 이유는 위에서 언급한 것처럼 데이터 방송 서비스를 제공하기 위해 콘텐츠 제작 업체가 사용할 수 있는 대역폭이 제한적이기 때문에 클래스 라이브러리를 포함하여 전송하였을 때 무리가 없는지를 테스트하기 위함이다. 여기서 전체 애플리케이션의 크기란 콘텐츠가 셋탑박스 상에서 구동되기 위해 필요한 모든 데이터를 합한 크기를 말하는 것으로 자바 클래스 파일 외에도 클래스 라이브러리 패키지, 이미지, 사운드 등 리소스 파일이 포함된다.

표 4를 통해서 알 수 있듯이 제안한 클래스 라이브러리를 통해 구현된 오목 콘텐츠는 전체 애플리케이션의 크기가 기존의 MHP 기반 오목 콘텐츠보다 32KB 컸다. 이는 본 논문에서 새로 작성한 오목 콘텐츠는 설계한 클래스라이브러리를 바탕으로 제작되어 있기 때문에 올바른 동작을 위해서는 설계한 클래스 라이브러리를 jar 파일 형태로 함께 전송해야 했기 때문이다.

표 4. 크기 및 코딩 라인 비교

비교 항목	기존 콘텐츠	새로 구현된 콘텐츠
전체 애플리케이션 크기	336 KB	368 KB
코딩한 클래스 파일의 크기	63.9 KB	45.4 KB
코딩 라인 수 (전체 클래스)	3660 line	2765 line
코딩 라인 수 (GameContainer 클래스)	1522 line	787 line

그러나 클래스 라이브러리, 이미지, 사운드 등을 제외하고 실제 개발자가 작성한 순수 자바 코드만을 포함하는 클래스 파일의 크기와 코딩 라인수를 비교한 결과 제안한 클래스 라이브러리를 통해 구현된 콘텐츠가 기존의 콘텐츠에 비해 약 30%정도 줄었음을 알 수 있다.

실제 게임 콘텐츠는 콘텐츠의 특성상 게임 자체의 알고리즘 표현이 콘텐츠의 많은 부분을 차지하고 있다. 따라서 GameAlgorithm 클래스에서 다음 돌을 놓는 위치를 결정하는 오목 알고리즘 구현 관련 부분이 중복될 수밖에 없음을 감안했을 때, 본 논문에서 작성한 클래스 라이브러리의 효율성을 판단하기 위해서 콘텐츠의 디스플레이적인 내용과 인터랙션 부분을 표현하는 GameContainer 클래스의 변화를 살펴보았다.

그 결과 GameContainer 클래스의 코딩 라인이 기존의 콘텐츠에 비해 약 50% 정도 줄어들어 코딩이 훨씬 간단해졌음을 알 수 있었으며, 전체 코딩 라인의 변화에 GameContainer 클래스의 변화가 가장 큰 영향을 미침을 알 수 있었다.

(4)로딩시간및응답시간

마지막으로 두 콘텐츠를 셋탑박스에서 구동하였을 때 애플리케이션 로딩 시간 및 게임진행에 따른 컴퓨터의 응답시간을 비교하였다. 로딩 시간 및 응답시간은 스트림스테이션에 표기되는 진행시간을 토대로 하여 총 10번의 테스트 후 평균을 산출하였다.

먼저 로딩시간을 비교해본 결과 기존 오목 콘텐츠는 로딩시간이 평균 15.6초, 새로 구현된 오목 콘텐츠는 평균 16.8초로 새로 구현한 오목 콘텐츠가 약 1.2초 정도 로딩 시간이 더 필요함을 알 수 있었다. 이는 실제로 콘텐츠를 접하게 되는 사용자는 거의 차이를 느끼지 못하는 정도의 시간 차이이며, 이처럼 로딩시간이 더 필요한 이유는 새로 구현한 콘텐츠가 클래스 라이브러리 패키지를 포함하고 있기 때문이다.

다음으로 응답시간을 측정하였다. 본 테스트에서 의미하는 응답시간이란 리모콘 입력에 따라 화면 디스플레이의 변화가 나타나는 시간을 의미하는 것으로 데이터 방송에서는 셋탑박스의 제한된 메모리 영역으로 인하여 디스플레이 타임 딜레이(time delay)가 발생하기 쉽다. 측정 결과 두 콘텐츠 모두 1초 이내의 반응을 보여 두 콘텐츠 간의 차이를 전혀 느낄 수 없었으므로 애플리케이션의 전체 용량이나 콘텐츠를 구성하고 있는 디렉토리의 뎁스(depth)가 응답시간에 미치는 영향은 미미함을 알 수 있었다.

5. 결 론

DTV에서의 데이터 방송은 상이한 미들웨어 규격으로 인해 콘텐츠 호환 및 재사용에 관한 문제가 발생하였으며 본 논문에서는 이러한 문제를 미들웨어 독립적인 클래스 라이브러리를 통해 해결하고자 하였다. 미들웨어 독립적인 클래스 라이브러리를 이용함으로써 데이터 콘텐츠의 재사용성이 증가하고 콘텐츠 개발자가 표준의 상이한 규격을 숙지할 필요가 없어 양질의 콘텐츠를 개발할 수 있는 환경 조성되어 국내 데이터 방송 콘텐츠 개발이 활성화 될 수 있으리라 기대한다.

또한 데이터 방송은 부가적인 멀티미디어 데이터를 전송함으로써 사용자에게 더욱 진보된 콘텐츠를 제공해주기 때문에 그 개념이 DTV에 국한되지 않고 차세대 방송인 DMB(Digital Multimedia Broadcasting)에서도 이용되고 있으며 앞으로 더 활발한 연구가 이뤄질 것이라 예상한다. DMB는 TDMB와 SDMB로 나뉘어져있을 뿐 아니라 DTV에 비해 훨씬 다양한 형태의 단말의 이용이 예상되므로 이러한 콘텐츠 호환의 문제는 DTV 뿐만 아니라 DMB 환경에서도 발생 할 수 있을 것이라 추측된다. 따라서 향후에는 DMB의 데이터 콘텐츠에 대한 연구와 함께 DMB 환경에서 미들웨어 독립적인 클래스 라이브러리의 응용에 관한 연구가 이뤄져야 할 것이다.

참 고 문 헌

[1] European Telecommunications Standards Institute (ETSI) Digital Video Broadcasting(DVB) ; Multimedia Home Platform(MHP) Specification 1.0.3, June 2003.

[2] OCAP, OpenCable Application Platform (OCAP) Specification: OC-SP-OCAP1.0-I11-040604, June 2004 .

[3] ATSC Standard A/101: Advanced Common Application Platform (ACAP), August 2005.

[4] 김경미, "ITU-R SG 6," *TTA 저널*, 제83호,

pp198~203, 2002년 10월.
[5] DVB-GEM, Digital Video Broadcasting; Globally Executable MHP specification, May 2004.
[6] DAVIC 1.4.1p9, DAVIC 1.4.1 Specification Part9, Complete DAVIC Specifications, June 1999.
[7] HAVi, HAVi specification version1.1, May, 2001.
[8] JavaTV, Java TV Specification Version 1.0, Sun Microsystems, November, 2000.
[9] JMF: Java Media Framework, Version 1.18, October, 1997.
[10] 최진수, 김진웅, "데이터 방송 잠정 표준 (TTAI.KO-07. 0015)," *TTA 저널*, 제77호, pp57~65, 2001년 9월.
[11] 정재훈, "데이터 방송용 게임 개발 및 채널 운영," *정보처리학회지* 제 11권 제 5호, 2004년 9월.
[12] 류주현, "디지털 케이블 방송 표준 및 미들웨어 기술," *정보처리학회지* 제 11권 제 5호, 2004년 9월.

임 현 정

2003년 숙명여자대학교 멀티미디어과학과(학사)
2005년 숙명여자대학교 멀티미디어과학과(석사)
2005년~현재 숙명여자대학교 응용
관심분야 : 디지털방송, DMB, Web3D, 멀티미디어

임 순 범

1982년 서울대학교 계산통계학과(학사)
1983년 한국과학기술원 전산학과(석사)
1992년 한국과학기술원 전산학과(박사)
1989년~1992년 (주)휴먼컴퓨터 창업 / 연구소장
1992년~1997년 (주)삼보컴퓨터 프린터개발부 부장
1997년~2001년 건국대학교 컴퓨터과학과 교수
2001년~현재 숙명여자대학교 멀티미디어과학과 교수
관심분야 : 컴퓨터 그래픽스, 웹 멀티미디어 응용, 디지털 방송, 전자출판(폰트, XML, 전자책, e-Learning)