논문-06-11-3-09

# 데이터방송 표준간 호환 가능한 저작도구 설계 및 구현

이 차 원[a], 남 윤 석[a], 김 정 환[a], 나 희 주[a], 김 성 원[a], 정 문 열[a]‡

# Design and Implementation of Data Broadcasting Authoring Tool for Multi-platforms

Cha Won Lee[a] , Yoon Seok Nam[a] , Jung Hwan Kim[a] , Hee Joo Na[a] , Seong Won Kim[a] and Moon Ryul Jung[a]‡


## 요 약

디지털 방송에서는 양방향 콘텐츠를 제작하기 위한 표준이 존재한다. 우리나라에서 지상파 방송은 ACAP(Advanced Common Application Platform), 위성 방송은 MHP(Multimedia Home Platform), 케이블 방송은 OCAP(OpenCable Application Platform)을 사용한다. 표준이 다르면 데이터방송을 위한 애플리케이션이 달라지고, 한번 만들어진 애플리케이션을 이종(異種) 매체를 통해 전송하려 할 경우 다시 만들어야하는 낭비가 발생한다. 본 논문에서는 이런 문제를 해결하기 위해 데이터방송 표준을 비교하여, 어떠한 API(Application Program Interface)들이 사용되는지를 파악하였다. 그리고 애플리케이션의 내용을 각 표준별 애플리케이션으로 변환 가능한 XML 파일로 표현하는 방법을 제안하였다. 또한 사용자와의 상호작용을 통해 XML 파일을 자동생성하는 저작도구를 구현하였다. 저작도구를 이용하여 제작된 Xlet 애플리케이션이 각 표준 환경에서 동일하게 작동하는지를 실험 검증하였다.

## Abstract

There are several data broadcasting standards that the developers should apply when develops any interactive digital TV contents. In Korea, we adopts ACAP(Advanced Common Application Platform) as terrestrial data broadcasting standard, MHP(Multimedia Home Platform) for satellite data broadcasting and OCAP(OpenCable Application Platform) for cable data broadcasting. Therefore, content providers must develop different applications which suited for different standards - even if the application is exactly same. This is waste of time and energy. This paper suggests a solution to solve this problem among different broadcasting platforms - MHP, OCAP and ACAP. We compared these specifications, and found out what APIs are typically used for applications. Then we designed XML file structures that can be used to define contents of applications. We also implemented an authoring tool which automatically generates XML files by interaction with users. This paper also ascertains Xlet applications, produced by the authoring tool, works properly on each different data broadcasting standard - MHP, OCAP and ACAP.


Key Words : Data Broadcasting, ACAP, OCAP, MHP, authoring tool

a) 서강대학교 영상대학원 미디어공학과
Digital Broadcasting Lab, Dept of Media Technology. Graduate School of Media Communications Sogang University
‡ 교신저자 : 정문열(moon@sogang.ac.kr)

## I. 서 론

2006년 지상파 데이터방송이 본방송에 들어감에 따라

데이터방송 콘텐츠 제작이 활발히 진행되고 있다. 정보를 제공해주는 독립형 서비스부터 시청자와 인터랙션이 가능한 연동형 서비스까지 다양한 콘텐츠가 제작되고 있다[1]. 이러한 콘텐츠를 제작하기 위해서는 각 서비스 매체별로 데이터방송 표준이 필요하고, 이를 바탕으로 콘텐츠를 제작하게 된다. 국내 데이터방송의 경우 지상파 방송은 ATSC(Advanced Television Systems Committee)-ACAP[2], 위성 방송은 DVB(Digital Video Broadcasting)-MHP[3], 케이블 방송은 CableLabs-OCAP[4]의 표준을 사용하도록 권고하고 있다. 하지만, 콘텐츠 개발사(CP) 입장에서는 이러한 매체별 표준의 차이로 인하여 콘텐츠의 재사용이 힘들며 각 매체에 맞게 콘텐츠를 재변환 하는 과정이 필요하게 되고, 개발 비용이 증가하는 문제점들이 발생한다[5].

이와 같은 매체별 표준의 상이함을 극복하고 OSMU (One Source Multi Use)를 실현하기 위해 본 논문에서는 각 표준에 따르는 애플리케이션으로 변환할 수 있는 XML 파일을 정의한다. 그리고 사용자와 상호작용하면서 이러한 XML 파일을 쉽게 작성하고, 이를 매체별 Xlet[6] 애플리케이션을 변환할 수 있는 저작도구를 구현한다. 또한, 애플리케이션의 전송은 방송 스트림을 통해 전송되는데, 비디오 및 오디오 신호가 방송 스트림을 통해 동시에 전달되기 때문에, 순수하게 데이터방송을 위해 할당된 대역폭에는 한계가 있다[7]. 그리고, 방송 스트림을 수신하는 셋톱박스는 PC보다 낮은 처리 능력을 가지고 있다. 즉, 양방향 데이터방송 콘텐츠가 원활하게 전송되고 개발자의 의도대로 구현하기 위해서는 전송하는 애플리케이션의 크기를 최대한으로 줄여야 한다. 따라서, 본 저작도구에서는 이러한 디지털 방송 환경을 감안하여 애플리케이션의 용량을 최소화 시킬 수 있는 방법에 대하여 논한다. 아울러 본 고에서는 위에서 언급한 데이터방송 표준을 제외한 ATVEF, ARIB, MediaHighway, OpenTV 등 국내에서 채택되지 않은 표준에 대해서는 다루지 않는다.

본 논문의 2장에서는 각 표준간의 비교 분석과 상호 호환이 가능한지 알아본다. 그리고 3장에서는 이를 바탕으로 생성되는 Xlet 생성에 대해 설명한다. 마지막으로 4장에서는 애플리케이션을 쉽게 제작할 수 있는 저작도구 구현에 대해 기술하고 결론을 맺는다.

## Ⅱ. 데이터방송 표준간 비교 분석

그림 1 에서 보듯이 MHP 1.0을 바탕으로 GEM(Globally

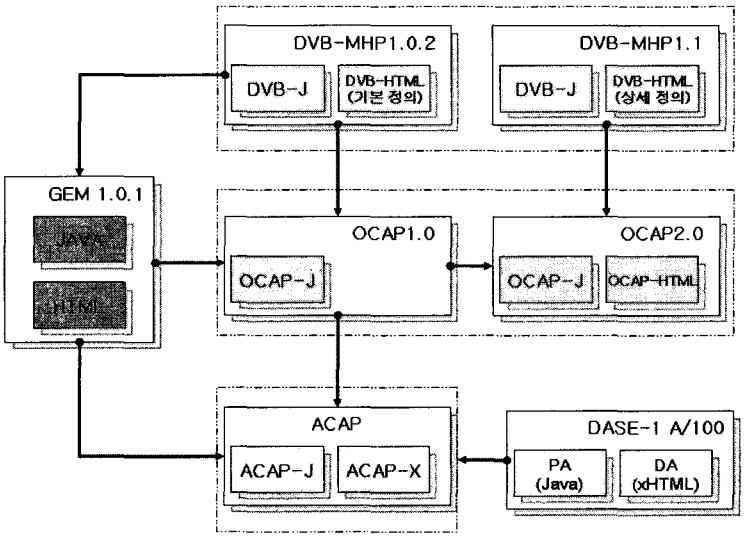



그림 1. 데이터방송 표준 버전별 상관도
Fig. 1. Relationship among MHP, GEM, and other standards.

Executable MHP)[8] 표준이 만들어졌고, GEM 표준과 MHP 표준을 바탕으로 OCAP 표준이 만들어졌다. 그리고, OCAP 표준과 DASE(Digital TV Application Software Environment)[9] 표준을 바탕으로 ACAP 표준이 만들어졌다. MHP, GEM, OCAP, ACAP의 순으로 표준이 제정되었지만 전자가 만들어지고 나서 후자가 만들어 진 것은 아니다. GEM은 유럽의 방송표준화 기구인 DVB가 정한 상위 표준으로, 하위의 기술규격들(ACAP · OCAP · MHP)은 GEM 표준을 따르고 있다. 하지만, GEM이라는 공통 표준을 준수하고 있다하더라도 명확히 기술되지 않은 부분들이 존재하므로, GEM이라는 공통의 표준을 바탕으로 각각의 표준이 정의 되어 있을뿐이라 할 수 있다.

## 1. 매체별 표준 API 특징

### 1.1 DVB-MHP

ETSI(European Telecommunication Standards Institute) 기관에서 채택한 유럽 지역을 대상으로 한 데이터방송 미들웨어 표준 규격으로서 현재 MHP 1.0 (DVB-J 미들웨어 규격) 과 MHP 1.1(MHP1.0+DVB-HTML)[10]에 대한 표준이 나와 있다.

### 1.2 CableLabs-OCAP

SCTE(Society of Cable Television Engineers)기관이 채택한 북미 케이블 방송 사업자를 위한 데이터방송 미들웨어 표준으로 MHP1.0을 기반으로 한 OCAP1.0과 MHP1.1을 기반으로 한 OCAP 2.0[11] 표준이 있다. OCAP에는 케이블 방송의 특성을 지원할 수 있는 다양한 세부 기술규약이 존재하는데, 예를 들어 양방향 서비스나 SI(Service Information)를 위한 서비스 프로토콜로 OOB가 정의되어 있다.

### 1.3 ATSC-ACAP

북미 텔레비전 협회에서 제정한 미국식 지상파 미들웨어 표준인 ACAP은 GEM을 기반으로 하여 케이블에서 제정한 OCAP과의 호환성을 보장하고 있으며, DAE(Declarative Application Environment)의 경우에는 선택사항으로 하고 있다. ACAP 표현 프로토콜로 ACAP-J(Java 환경)와 ACAP-X(Extensible Hyper Text Markup 언어 환경)로 구성되어 있으며, Cable Network를 위한 Monitor Application 지원 등의 특징을 가지고 있다.

## 2. 매체별 표준 API 구성

DVB-MHP, CableLabs-OCAP, ATSC-ACAP 데이터방송 표준 API는 공통적으로 Personal Java 1.2[12], Java TV 1.0[6], JMF(Java Media Framework) 1.0[13], JSSE(Java Secure Socket Extension) 1.02[14], HAVi(Home Audio Video interoperability) Level 2 User Interface[15], DAVIC (Digital Audio Video Council) 1.4.1[16] API를 채택하고 있다. 그리고, 공통 API 와 더불어 각각의 매체별 특징을 반영한 Specific API를 사용하고 있다. 하지만 공통적으로 채택한 API 일지라도 매체별 특징을 반영하기 위해 제외, 대체되는 부분이 있으며, 매체별 Specific API를 타 매체에서 채택하여 미들웨어 플랫폼을 구성하는 경우도 있다.

## 3. 매체별 표준 API 비교 분석

어떤 플랫폼에서든지 상호 호환 가능한 애플리케이션을 제작하기 위해서는 매체별 표준 API의 비교가 선결되어야만 한다.

### 3.1 JMF, JSSE, JavaTV, HAVi Level 2 UI API

JMF는 비디오와 오디오를 제어할 수 있도록 하는 API이다. 그리고, JSSE는 리턴채널의 암호화 통신을 위한 SSL API이다. 또한, JavaTV는 셋톱박스 내에서 애플리케이션과 RTOS의 사이에 존재하는 API로 애플리케이션이 실행 가능하도록 해주는 역할을 한다. 마지막으로, HAVi는 디지털 가전 내부의 기능 구현을 위한 API로써 데이터방송에서는 TV와 친숙한 UI를 구성하기 위해 UI 관련 API만 사용한다. 이러한 API들은 DVB-MHP, CableLabs-OCAP, ATSC-ACAP 데이터방송 표준에서 공통적으로 지원한다.

### 3.2 Personal Java 1.2

JDK 1.1.8 API의 대부분을 수용하고 있지만 TV의 성격에 맞지 않는 몇 가지 부분이 제외되어 있다. 매체별 공통적으로 com.sun, java.applet, java.sql, java.text 패키지들은 이러한 이유로 매체별 표준 API에서 제외되어 있다. DVB-MHP 표준에서는 java.awt 패키지 중 heavy weight 컴포넌트, java.beans, java.math, java.net 패키지에 대하여 부분적으로 수용한다고 명시되어 있다.

### 3.3 DAVIC API

- org.davic.media 패키지
  SubtitlingLanguageControl 인터페이스는 OCAP, ACAP에서 사용하지 않으며 대응되는 것으로 Closed CaptioningControl 인터페이스를 들 수 있지만 그 구성 요소가 다르다.
- org.davic.mpeg.dvb 패키지
  DVB전송방식의 MPEG 개념을 위한 유틸리티로서 MHP에서 지원된다. OCAP, ACAP은 좀 더 일반화되고 대응되는 org.davic.mpeg 패키지를 지원한다.
- org.davic.net.ca 패키지
  CAS(Conditional Access System)에 관련된 것으로서 매체별 하드웨어에 의존도가 높다. OCAP은 CA (Conditional Access) 인증이 POD(Point of Deployment)에 의하여 수행되므로 org.davic.net.ca 패키지를 지원하는 MHP와 차이가 있다.
- org.davic.net.dvb 패키지
  DVB에 특화된 콘텐츠의 참조를 제공하는 것으로 MHP에서 이 패키지를 지원한다. OCAP 과 ACAP에서는 org.davic.net.dvb 패키지의 DvbLocator 클래스 대신 org.davic.net 패키지의 Locator 클래스와 org.ocap.net 패키지의 OcapLocator, OCRCInterface 클래스를 지원하고 있다.

### 3.4 org.dvb API

- org.dvb.media 패키지
  JMF를 위한 DVB Specific 확장으로 자막과 CA에 관련된 인터페이스와 클래스는 OCAP, ACAP에서 지원하지 않으며, 이에 대응되는 OCAP, ACAP Specific 패키지를 지원하고 있다.
- org.dvb.net.tuning 패키지
  DAVIC의 tuning API를 DVB에 맞게 확장한 것으로 OCAP과 ACAP에서는 DVB의 SI를 지원하지 않으므로 DvbNetworkInterfaceSIUtil 클래스를 지원하지 않는다.
- org.dvb.si, org.dvb.net.ca 패키지
  DVB에서 정의한 Specific한 API로 OCAP과 ACAP에서는 지원되지 않고 있으며, 매체별 특성에 대응되는 알맞게 Specific API형태로 각각의 표준에서 재정의하고 있다.

### 3.5 org.ocap API

org.ocap API은 GEM을 상위 규격으로 케이블 환경에 맞게 제정된 API이다. 가장 큰 특징으로 OOB 서비스 프로토콜과 POD, 그리고 Monitor application을 들 수 있다. MHP는 이러한 개념들이 없기 때문에 org.ocap API를 지원 하지 않고 있으며, ACAP은 Cable Network을 위해 일부분을 채택하고 있다.

### 3.6 org.atsc API

org.atsc API는 아래 표 1 에서 보듯이 Specific API 로써 다른 표준들은 지원하지 않는다.

표 1. org.atsc API 표준별 비교표
Table 1. Table of org.atsc API Specification

| Packages | ACAP | OCAP | MHP |
|---|---|---|---|
| org.atsc.si | 지원 | 지원안함 | 지원안함 |
| org.atsc.dom | 조건부지원 | 지원안함 | 지원안함 |
| org.atsc.dom. environment | 조건부지원 | 지원안함 | 지원안함 |
| org.atsc.dom.event | 조건부지원 | 지원안함 | 지원안함 |
| org.atsc.dom.html | 조건부지원 | 지원안함 | 지원안함 |
| org.atsc.dom.views | 조건부지원 | 지원안함 | 지원안함 |

### 3.7 표준간 상호 호환성

위에서 살펴본 봐와 같이 MHP, OCAP, ACAP 표준들 간

의 서로 상이한 API는 Specific API (org.dvb.*, org.ocap.*, org.atsc.*)를 제외하면 거의 존재하지 않으며, 설사 있다고 하더라도 대응되는 API를 통해 쉽게 변환이 가능하다. 하지만, 매체별 Specific API (DVB 프로토콜을 위해 정의한 API, CA, 케이블환경의 OOB, POD, SI 등)는 매체 별로 정확하게 대응되는 API가 명확히 기술되어 있지 않다. 따라서 매체 별로 각기 다른 환경에 따라 그에 걸 맞는 소스 코드를 저작 도구가 자동으로 생성하는데 어려움이 있다. 따라서 본 연구에서는 저작자가 Specific한 API를 직접 다룰 수 있도록, 자동 생성된 소스 코드의 수정 및 추가 기능을 제공 할 것이다.

## Ⅲ. Xlet 생성기 설계 및 구현

본 연구에서 구현한 Xlet 생성기를 이해하려면, Xlet 애플리케이션의 구조에 대한 이해가 반드시 필요하다. 따라서, 이 장에서는 먼저 기본적인 Xlet 애플리케이션의 구조에 대해 설명한 뒤, Xlet 생성기의 구조를 크게 XML 파일 생성, Xlet APIs, Xlet 생성부의 세 가지의 중심 요소로 나누어 설명할 것이다.

### 1. Xlet 애플리케이션의 구조

Xlet 애플리케이션은 ACAP, OCAP, MHP 셋톱박스에서 실행되는 Java 애플리케이션이라 하는데, 이것은 Downloadable 한 애플리케이션으로써 애플릿이 웹 브라우저에 다운로드되어 돌아가듯, 방송 전파를 통해 셋톱박스에 다운로드되어 수행되는 자바 프로그램을 말한다. Xlet은 Applet 처럼 Life Cycle을 가지고 있다[17]. Loaded, Paused, Active, Destroyed의 네 가지 상태가 있으며 각 단계의 호출에 의해 Xlet의 상태가 변화된다.

Loaded 상태는 Xlet이 load되었지만 아직 초기화 되지 않거나 혹은 일부만 초기화된 경우이다. Application manager가 Xlet의 새로운 인스턴스를 만들게 된다. 이 상태에서 만약에 exception이 발생하였다면 Xlet은 바로 Destroyed 상태로 넘어가게 되고 이 Xlet은 버려지게 된다.

Paused 상태는 Xlet은 초기화가 되었지만 어떠한 resource도 가지고 있지 않은 상태이고 Application manager가 Xlet을 위해 필요한 context 오브젝트를 생성한다. Loaded 상태에서 initXlet(), Active상태에서 pauseXlet(), 또 XletContext의 notifyPaused() method가 성공적으로 호출되면 이 상태로 넘어간다.

Active 상태는 Xlet이 모든 서비스를 제공하는 상태이며, 이 상태는 Paused 상태에서 startXlet()의 호출을 통해서만 넘어올 수 있다. 물론 Application manager에 의해 적절한 시간에 서비스가 이뤄질 것이라는 것을 검증 받아야 한다.

Destroyed 상태는 모든 resource를 놓고 Xlet을 종료하는 상태이다. 각 상태에서 destroyXlet() method의 호출로 destroy될 수 있다.

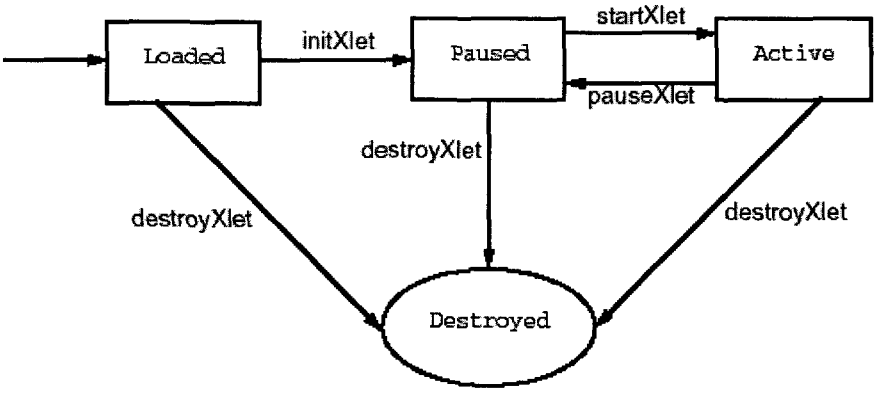



그림 2. Xlet 상태 전환표
Fig. 2. State Diagram of Xlet

## 2. XML 파일 설계

XML 파일은 Xlet 애플리케이션을 화면(Scene) 단위로 나누어서, 복수 개의 Scene을 생성 할 수 있도록 되어 있다. 그림 3은 XML 파일의 예로서 하나의 Scene 엘리먼트 밑에 여러 개의 SceneDef 엘리먼트를 둘 수 있으며 또한, 그 하위에 여러 개의 Properties 엘리먼트를 정의 할 수 있다.

### 2.1 Application 엘리먼트

Application 엘리먼트는 XML의 최상위 엘리먼트로 하나의 Xlet Application을 가지게 된다. Application 엘리먼트의 속성값으로는 name과 type이 있다. name 속성은 Application 엘리먼트의 이름으로 추후 생성시 Xlet 클래스명으로 쓰이게 되며, 송출시 설정하는 Init Xlet 클래스명으로 사용되는 중요한 속성이다. Type 속성은 Xlet이 특정 플랫폼에 맞게 변환되도록 사용하는 값이다. 각 표준에서 정의하는 콘텐츠 화면 해상도로부터 특화된 API를 사용하기 위해서는 저작자가 플랫폼을 선택해야 한다. 본 논문에서 제안하는 저작도구는 이 속성을 Xlet 생성부에서 인식하여 각 플랫폼에 적합한 Xlet을 생성하도록 구성되었다.

### 2.2 Scene 엘리먼트

Application 엘리먼트의 자식 엘리먼트로써 여러 개의 Scene 엘리먼트들이 구성될 수 있다. 각각의 Scene 엘리먼트는 방송 화면을 구성하는 가장 기본적인 단위이며, 최소 하나 이상의 Scene을 생성하게 된다. Scene 엘리먼트는 속성으로 name을 가지고 있으며, 이 name은 각 Scene을 구성하는 자바 소스파일의 클래스명으로 사용되게 되므로 Application 엘리먼트의 name과 중복되지 않아야 하며, 다른 Scene 엘리먼트의 name과도 동시에 사용 하면 안된다.

### 2.3 SceneDef 엘리먼트

Scene 엘리먼트의 자식 엘리먼트로써 위의 Scene 엘리먼트는 바탕을 의미하고 실제 바탕 안에 구성되어 있는 요소들을 정의하는 엘리먼트가 SceneDef이다. 또한, 이 엘리먼트는 Xlet Application 에서 가능한 기능들을 type 속성을 통해 분류된다. type들의 설명은 다음과 같다.

■ Image type

화면에 이미지를 보여줄 때 사용되는 type이다. name 속성은 이미지 파일의 명칭을 나타내며 X, Y, Width, Height 는 이미지의 좌표 값과 크기의 속성을 나타낸

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Application name="TestXlet" type="OCAP">
  - <Scene name="MainScene">
      <SceneDef name="image3414" type="IMAGE" x_value="34" y_value="14" height="444"
        width="681" abspath="C:\pg_ui2.png" path="images/pepg_ui2.png" />
      <SceneDef name="resize1" type="RESIZEVIDEO" x_value="380" y_value="50" height="306"
        width="321" />
    - <SceneDef name="text8855" type="TEXTBOX" x_value="88" y_value="55" width="222"
        height="29" fill="true">
        <Properties type="TextColor" red="255" green="255" blue="255" alpha="255" />
        <Properties type="Font" name="Dialog" size="20" style="BOLD" />
        <Properties type="String" align="CENTER" text="테스트 화면 입니다" />
      </SceneDef>
    </Scene>
  - <Scene name="SecondScene">
    - <SceneDef name="text432391" type="TEXTBOX" x_value="432" y_value="391" width="0"
        height="29" fill="true">
        <Properties type="TextColor" red="0" green="0" blue="0" alpha="255" />
        <Properties type="Font" name="Dialog" size="20" style="PLAIN" />
        <Properties type="String" align="CENTER" text="두번째 화면입니다" />
      </SceneDef>
    </Scene>
</Application>
```

그림 3. XML 파일의 구조

Fig. 3. The Structure of XML files

다. 그리고, path 는 이미지 파일 경로 속성을 가진다.

- TextBox type
  화면에 글자가 들어 있는 상자를 표현할 때 사용되는 type이다. 기본적인 속성은 Image type 과 동일하고, 글자와 상자에 대한 세부 속성을 정의하기 위해 자식 엘리먼트인 Properties 엘리먼트를 이용하여 정의한다.
- Rect type
  사각형을 그리는 type이다. 필요한 속성 값은 위와 동일하고 추가적으로 fill 속성을 부여하여 사각형의 색을 채울지 안채울지 결정한다. 그 색에 대한 속성은 Properties 엘리먼트를 이용하여 정의한다.
- Resize type
  비디오 영상의 크기를 조절할 때 사용되는 type이다. 크기 값을 이용하여 사이즈를 조절하게 된다.
- Sound type
  사운드 파일(mp2)를 재생하기 위해 사용되는 type이다. 사운드 파일의 위치 값을 속성으로 갖는다.
- IFrame type
  배경화면으로 사용되는 MPEG I-Frame 을 나타내기 위한 type이다. 속성 값으로 I-Frame 파일 위치 값을 갖는다.
- Input type
  사용자 입력을 받을 수 있도록 하는 type이다. 속성 값으로는 위치 값과 입력받을 수 있는 최대 길이가 있으며, 입력 상자의 색에 대한 세부 속성은 Properties 엘리먼트를 이용한다.

### 2.4 Properties 엘리먼트

SceneDef 엘리먼트에서 가지고 있는 속성 중에서 좀 더 세부적인 속성이 필요할 때 쓰이는 엘리먼트이다. Scene Def type 속성에 따라 Properties 엘리먼트의 사용 여부가 결정 된다. 주로 Color 속성 이나 Font 설정에 관련된 엘리먼트를 설정하는데 사용한다.

## 3. Xlet APIs 설계 및 구현

위에서 언급했던 SceneDef 엘리먼트의 type 속성에 정의된 기능을 Xlet에서 구현되도록 준비된 클래스들의 집합을 Xlet APIs라고 한다. 이는 Xlet에서 사용하는 기본적인 기능들을 추상화하여 단순한 기능부터 플랫폼에서만 정의되어 있는 기능들을 API화하여, Xlet 생성부에서 이를 쉽게 가져다 사용할 수 있도록 하였다[18]. 각각의 Scene을 관리하는 매니저를 기반으로 GUI 관리 매니저, 비디오/사운드 관리 매니저, 그래픽관련, 백그라운드 매니저로 구성 되어 있다. 또한, 각각의 매니저가 항상 생성되는 Xlet과 함께 전송이 되어야 하는 것이 아니라 XML 파일에서 사용된 기능들의 클래스만 같이 전송 되어 전체 Xlet 용량의 감소와 Xlet 로딩시간을 단축시킬 수 있는 이득이 있다.

### 3.1 SceneManager 클래스

Scene과 Xlet을 관리하며 전체적인 애플리케이션을 관장하는 가장 중요한 클래스이다. 이는 SceneDef 엘리먼트의 type 속성과는 상관없이 Xlet이 생성 되면 항상 같이 전송되어야 하는 기본 클래스를 의미한다. 기본적으로 Xlet이 가지고 있는 Life Cycle을 위임받아 자원을 관리하는 기능을 하게 된다[19]. 또한, Application 엘리먼트 속성에서 type을 받아들여 플랫폼 특성에 맞는 Xlet 으로 운영될 수 있도록 설계가 되어 있다. 그리고, Scene을 관리하여 화면에 보여지는 Scene에 대하여 제어 할 수 있으며, 생성되는 Scene 클래스들을 동적으로 로딩하여 관리하도록 되어 있다. 또한, MediaManager 클래스와 연계하여 화면 사이즈 조절(Resize type) 과 사운드 파일 재생(Sound type)이 가능하도록 되어 있다.

### 3.2 DefaultScene 클래스

복수 개의 Scene이 생성되면 그들의 클래스들은 Default-Scene 클래스을 상속받아 구현되어 있게 된다. 이 클래스는 Scene을 구성하는 가장 기본적인 요소들을 정의해놓은 추상 클래스인 것이다. HContainer 클래스를 상속받아 Container 클래스의 속성을 따르며, KeyListener 인터페이스를 구현하여 사용자 이벤트를 처리할 수 있도록 설계 되어 있다. 또한, Scene이 생성될 때 이를 SceneManager 클래스에게 알려 Scene을 관리할 수 있도록 되어 있다. 그리고, 추상 클래스로써 기본적인 요소만 정의해 두었고 그 외

사용자의 이벤트 처리 부분은 추상 메소드화 하여 저작자가 직접 입력을 할 수 있다.

### 3.3 GUIManager 클래스

Image 및 기타 UI 관련 기능의 구현 및 제어를 담당하는 클래스이다. Image type, TextBox type, Input type을 구현하였으며 또한, HAVi에서 제공하는 메소드를 이용하여 UI 형태가 TV 화면에 적합한 모습처럼 보이도록 하였다.

- Image type
  HIcon 클래스를 이용하여 이미지에 대해서 HIcon 파라미터에 따른 속성 값을 주고 화면에 보이도록 하였다. 이미지 로딩은 MediaTracker 클래스를 이용하였으며, 이미지 로딩에서 많은 자원과 시간이 소요되므로 사용이 끝난 후에는 항시 자원을 회수하도록 되어 있다.
- TextBox type
  HStaticText 클래스를 이용하여 HStaticText에서 사용되는 속성 값을 파라미터로 받아들여 적용시켰으며, 이를 오버로딩하여 파라미터 개수에 따른 적용을 다양하게 받아 들일 수 있도록 하였다.
- Input type
  HStaticText 클래스를 이용하여 입력받을 수 있을 크기의 배열로 생성하도록 하였다. 또한 입력받은 이벤트를 화면에 나타내기 위한 이벤트 처리를 하였다.

### 3.4 MediaManager 클래스

비디오와 사운드에 관련된 기능을 관리하는 클래스로써 사이즈 조절이나 사운드 재생을 하기 위한 초기화 작업이 이루어진다.

### 3.5 GraphicRect 클래스

이 클래스는 다른 클래스들처럼 관리하는 기능이 아닌 사각형을 그리기 위해 구현된 클래스이다. 컴포넌트를 상속받아 paint(Graphics g) 메소드에서 사각형을 그리도록 구현된 클래스이며, GUIManager 클래스에서 생성된 Rectangle 컴포넌트를 반환하는 방식을 취한다.

### 3.6 BackgroundManager 클래스

화면을 구성하기 위한 백그라운드 디바이스를 설정하고 I-Frame을 로딩할 수 있도록 하는 클래스이다. I-Frame 의 로딩은 시간이 오래 걸리고 자원 소모도 큰편이기에 이를 효율적으로 로딩할 수 있도록 자원 관리에 중점을 두었다.

## 4. Xlet 생성부 설계 및 구현

2, 3절에서 살펴보았던 기능들은 실제 Xlet 생성시 필요한 정보와 기능이며, 이를 바탕으로 실제 Xlet 이 생성되는 과정에 대해 알아보겠다.

### 4.1 Xlet 생성 방법

먼저, XML 파일을 분석하여 하나의 Scene 엘리먼트에서 필요한 SceneDef 엘리먼트의 type 속성들을 찾아낸다. 그다음, type 속성에 알맞은 Xlet APIs에서 정의된 메소드를 호출하는 문장 찾는다. 그리고, 이를 Xlet.Template 파일에 추가를 시켜 Xlet 코드를 완성시킨다. 위와 같이 하기 위해서는 각각의 type 별 메소드를 호출하는 방법을 정의한 규칙이 있어야 하고 이를 추가시킬 수 있는 기본적인 소스 파일 즉, Template 파일이 존재해야 한다. 그러므로, 기본 Xlet Template 과 Scene Template 파일 두 개가 존재하게 된다.

### 4.2 Xlet API 삽입 규칙

Template 파일에 메소드 호출 정보를 삽입할 위치는 총 네 곳이 있다. 삽입하는 부분은 각각 목적에 맞게 정의되어 있으며 SceneDef 엘리먼트의 type 속성별로 네 곳을 다 사용하여 삽입하는 경우가 있으며 두 곳만 사용하여 삽입하여 Xlet 소스 파일을 완성시키는 수도 있다. 이러한 각각의 삽입 위치와 삽입 문장을 정의해놓은 파일을 Template.prop 라 하며, 이 파일 안에는 이러한 규칙들을 저장해두어 생성시 참조하도록 되어 있다.

- Import 부분
  자바에서는 소스 외부의 클래스를 참조할 경우 Import 키워드를 사용하여 외부 클래스의 위치를 명시해야 한다. 이에 Xlet APIs 패키지를 사용하기 위해서는 Import 부

분에 type에 알맞은 클래스명을 삽입시켜야 한다.

- 선언 부분
  클래스 전체에서 객체를 참조할 경우에는, 클래스 멤버 변수 선언 부분에 선언을 하게 된다. 클래스 내부 메소드에서 이러한 객체를 사용할 때 참조 할 수 있게 되는 것이다.
- 생성자 부분
  선언 부분에서 선언된 객체를 실제 사용하기 위해서는 생성자 부분에서 객체 생성을 해주어야 한다. 또한 생성시 객체의 초기화도 동시에 이루어져서, 사용하고자 하는 클래스를 알맞게 생성하도록 하는 기능을 한다.
- 메소드 호출 부분
  위 세 군데 삽입 방법은 메소드 호출를 하기 위한 준비 작업이었다. 실제 메소드를 호출하기 위해 이 삽입이 이루어져야 한다. 위치는 기본적으로 Scene 클래스에서 init() 메소드내에서 이루어진다.

### 4.3 Xlet 생성

#### 4.3.1 기본 Xlet 변환

그림 4의 기본 Xlet.template 파일은 변환과 삽입을 통하여 Xlet 소스로 생성 된다. $xletName 이라고 명시되어 있는 부분은 XML 파일 Application 엘리먼트의 name 속성 값으로 치환된다. 생성자 부분에서는 생성되는 복수 개의 Scene 클래스를 SceneManager 클래스에 등록시키는 메소드를 삽입한다. 그리고, 표준별로 생성이 되도록 파라미터(MHP,OCAP,ACAP)를 입력받아 플랫폼에 맞게 생성되도록 한다.

XML 파일 Application 엘리먼트의 name 속성을 "TestXlet" 으로 주고 type 속성을 "OCAP" 으로 적용한 Xlet 으로 변환하게 되면 그림 5와 같은 결과가 나타난다.

#### 4.3.2 Scene 클래스 생성

기본 Xlet 생성과 마찬가지로 Scene 클래스에서 $sceneName 값은 Scene 엘리먼트 name 속성 값이 들어오게 된다. 그리고, Xlet 삽입 방법 규칙에 따라 SceneDef 엘리먼트의 type 속성에 맞도록 삽입이 이루어지게 되는 것이다.

## Ⅳ. 저작도구 구현 및 실험

본 연구에서 구현한 저작도구는 Xlet 프로그래밍을 잘하지 못하는 기획자, 디자이너 등이 손쉽게 양방향 데이터 방

```
Public class $xletName implements Xlet {
        private SceneManager sceneManager = null;
        private Vector sceneVector;
        public $xletName() {
                sceneVector = new Vector();
                this.sceneManager = new SceneManager(sceneVector,$type);      }
```

그림 4. Xlet.template 파일 부분 소스
Fig. 4. Xlet.template file

```
public class TestXlet implements Xlet {
        private SceneManager sceneManager = null;
        private Vector sceneVector;
        public TestXlet() {
  sceneVector = new Vector();
   sceneVector.add(TestScene1.class);
   this.sceneManager = new SceneManager(sceneVector,"OCAP");      }
```

그림 5. 생성된 Xlet 소스
Fig. 5. Created Xlet

송 콘텐츠를 저작할 수 있도록 하는 GUI 툴을 제공한다. 그리고, 본 저작도구는 앞서 언급한 API 분석을 토대로 제작하였고, 저작도구에서 생성한 XML 파일은 Xlet 애플리케이션에서 자주 사용하는 기능들을 API로 정의하여 사용한다. 또한 생성된 XML 파일을 각 매체 별 애플리케이션으로 변환하는 기능을 가지고 있다. 따라서 본 저작도구를 사용하여 애플리케이션을 저작할 경우, 플랫폼에 따른 API의 차이를 저작도구에서 자체적으로 수정해주므로, 각기 다른 플랫폼에서 실행 가능한 애플리케이션을 생성하는 것이 가능하다.

## 1. 저작도구 구조 및 기능

### 1.1 저작도구 구조

그림 6에서 보듯이 저작도구 구조는 크게 저작도구와 Xlet 생성기로 나눠진다. 저작도구에서 비쥬얼 편집기로 저작된 화면을 XML 파일로 생성하며 이를 Xlet 생성기로 전달하여 Xlet을 생성하게 된다.[20]

### 1.2 저작도구 기능

저작도구의 기능은 다음과 같이 크게 3가지로 나눌 수 있다.

- 비주얼 편집기에서 사용자의 입력을 받아 이를 XML 파일로 변환하는 XML 파일 생성 및 저장 기능
- XML 파일을 Xlet 생성기에 전달하여 이를 분석하여 Xlet 소스 코드를 생성 시키는 작업을 수행하는 기능
- 소스 코드를 컴파일 하여 클래스 파일을 생성하고, Xlet 실행에 필요한 클래스 파일의 삽입 및 각종 자원(Text, Image)파일을 모아 프로젝트 폴더를 생성 하는 기능

그리고, 사용자 이벤트 처리 입력을 저작도구 상에서 직접 수정 할 수 있도록 소스 에디트 창을 제공한다.

## 2. 저작도구 구현

저작도구는 PC 기반의 JAVA 프로그래밍 언어(JDK 5.0)로 개발 되었으며, 개발 툴은 공개용 소프트웨어인 Eclipse 3.1를 사용하였다. 공개 GUI 프레임 워크인 "JHotDraw 5.3"[21]을 이용하여 기본적인 그래픽 유저 인터페이스를 구현 하였다. "JHotDraw"는 도형 편집과 GUI 프레임워크로써 공개용이지만, 상업용 수준의 규모와 품질을

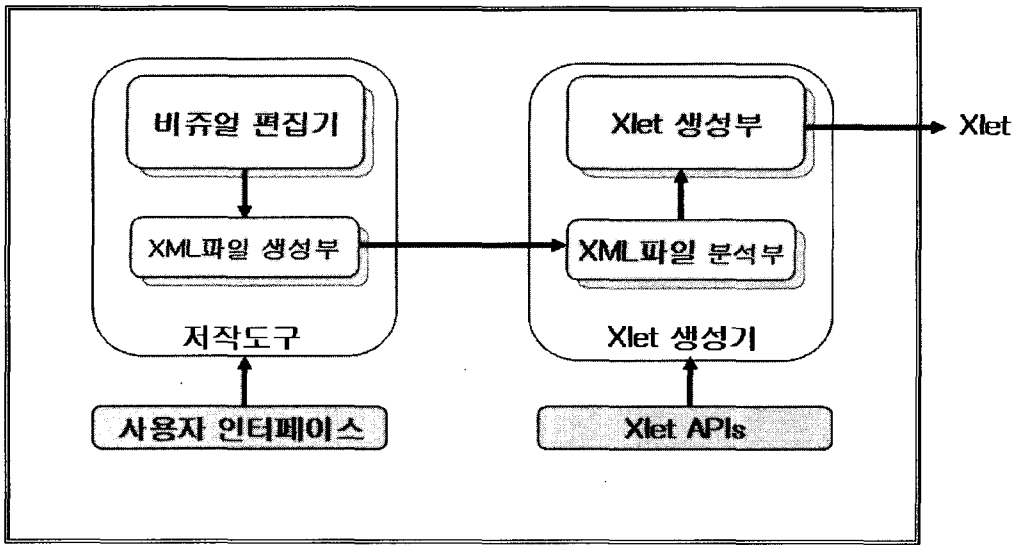



그림 6. 저작도구 구조도
Fig. 6. Structure of the Authoring Tool

갖추었다고 할 수 있을 정도의 체계적이고 안정적인 GUI 프레임워크를 제공한다.

## 3. 저작도구 실험

### 3.1 콘텐츠 제작

그림 7은 본 연구에서 구현 한 저작도구를 통해 간단한 EPG 화면을 구성해 본 것이다. 실험용 콘텐츠는 각 매체별로 전체 화면 크기와 비율의 차이에 대한 (ACAP : 640 X 480, OCAP : 720 X 480, MHP : 720 X 480) 문제를 저작도구에서 각 매체별 비율에 맞게 정확하게 변환해 주는지 검증하기 위해서 제작하였다. 저작도구 실험 환경은 인텔 펜티엄4 CPU가 장착된 PC를 사용하였으며, 운영체제는 Windows XP를 사용하여 실험 하였다.

### 3.2 콘텐츠 실험

본 논문에서 저작된 데이터방송 콘텐츠는 각 데이터방송 표준에 맞는 API를 이용하여 생성되었으므로, 제작된 콘텐츠는 각 표준을 지원하는 수신기에서 실행 되어야 한다. 본 저작도구를 이용하여 제작한 콘텐츠를 실제 셋톱박스로 전송하여 동작 실험을 하였다. 실험 셋톱박스는 OCAP 1.0 표준을 지원하는 "HUMAX OC-2500" 셋톱박스와 MHP 1.0 표준을 지원하는 "현대디지털테크 HSS-2100SP" 셋톱박스를 이용하였으며 ACAP 표준을 지원하는 셋톱박스는 아직 상용화되지 않아 실험 하지 못했다.

그림 8 에서 보듯이 저작도구에서 저작한 콘텐츠가 실제 셋톱박스에서 동작하는 것을 볼 수 있다. 이 화면은 OCAP 셋톱박스에서 구동한 화면이며 MHP 셋톱박스에서도 동일

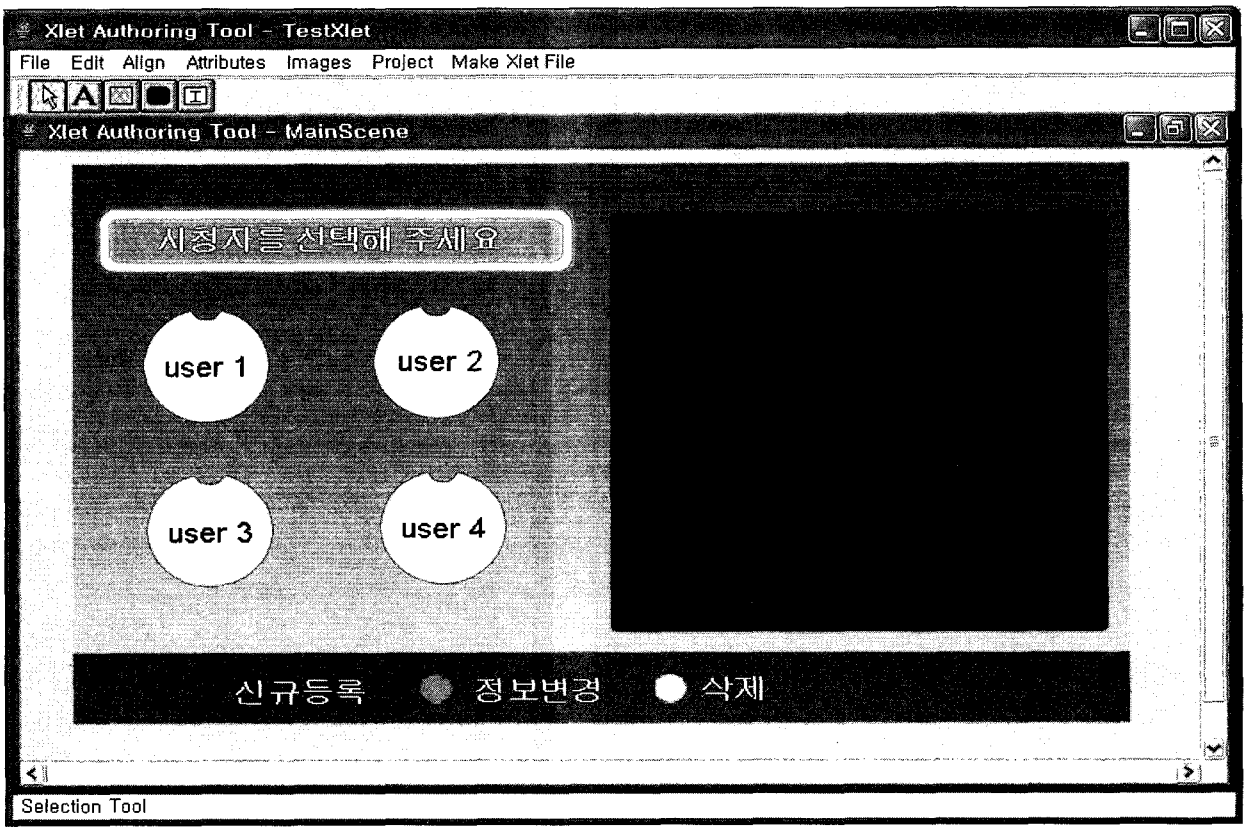



그림 7. 저작도구를 이용한 저작과정
Fig. 7. Work space of the Authoring tool


그림 8. 셋톱박스 실험 화면
Fig. 8. Actual demo application via STB

하게 동작하였다.

## V. 결 론

현재 디지털 방송은 급속히 양방향 데이터방송을 중심으로 발전하고 있다. 하지만, 각 매체에서 채택한 표준이 각기 달라 콘텐츠 제작사는 콘텐츠를 매체별로 중복 제작해야하는 불편이 생기게 되었다.

따라서 본 연구에서는 데이터방송 표준 상호간의 호환성을 분석하여, 이를 바탕으로 XML 파일을 설계하였다. 또한 XML 파일이 실제 Xlet 코드에 적용될 수 있도록 Xlet APIs 모듈을 제작한 뒤, 이를 저작도구로 구현하였다. 본 연구에서 제작한 저작도구를 사용하면 각각의 데이터 방송 표준에 적용 가능한 콘텐츠를 손쉽게 생성 할 수 있다. 그러나 상호 호환이 불가능한 매체 Specific API에 대해서는 공통 APIs 을 설계할 수 없는 문제점이 존재하므로, 이를 보완하기 위해 본 저작도구에서는 저작자가 자동 생성된 Xlet 코드에 Specific한 API 코드를 직접 입력하여 사용할 수 있도록 하는 소스 수정 창을 제공하였다.

또한, 본 연구에서 정리한 매체 별 데이터 방송 표준 분석은 향후 GEM을 위시한 데이터 방송 표준의 통합에 유용하게 사용할 수 있을 것이다. 아울러 함께 개발한 저작도구는 Xlet 프로그래머의 도움 없이 디자이너가 손쉽게 Xlet 애플리케이션을 제작할 수 있는 GUI 환경을 제공한다. 이를 통해 수많은 데이터방송 콘텐츠 개발사(CP)에서 보다 손쉽게 양방향 데이터방송 콘텐츠 제작이 가능해 질 것이다.

추후 전문적인 방송용 저작도구로 사용되기 위해서는 좀

더 다양한 그래픽 기능과 기존 방송 매체 도구의 연동이 필요하다. 또한, 실제 방송환경에서의 다양한 검증을 통해 지속적으로 기능개선이 이루어져야 할 것이다.

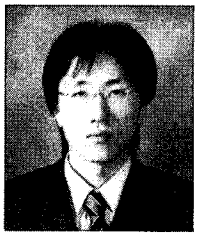
## 참 고 문 헌


[1] 김원용, 윤은상, "데이터 방송," 커뮤니케이션북스, 2판, 2005.
[2] ATSC Standard A/101, "Advanced Television Systems Platform," August, 2005.
[3] Digital Video Broadcasting (DVB), "Multimedia Home Platform 1.0.3," ETSI TS 101 812 V1.3.1, June 2003.
[4] ANSI/SCTE 90-1, "SCTE Application Platform Standard OCAP 1.0 Profile," 2005.
[5] 김대수, "국내 데이터방송 서비스 시장 동향," SW Insight 정책리포트, 14호, 2006.
[6] Sun Microsystems, Inc., "Java TV API 1.0 Specification," Nov 2000.
[7] Steven Morris, Anthony Smith-Chaigneau, "Interactive TV Standards," Focal Press, 2005.
[8] Digital Video Broadcasting (DVB), "Globally Executable MHP 1.0.2," ETSI TS 102 819 V1.3.1, Sept 2005.
[9] ATSC Standard A/100, "DTV Application Software Environment - Level 1 (DASE-1)," March 2003.
[10] Digital Video Broadcasting (DVB), "Multimedia Home Platform 1.1.1," ETSI TS 102 812 V1.2.1, June 2003.
[11] ANSI/SCTE 90-2, "SCTE Application Platform Standard OCAP 2.0 Profile," 2005
[12] Sun Microsystems, Inc., "Personal Java API 1.2 Specification," 1999.
[13] Sun Microsystems, Inc., "Java Media Framework 1.0 API Specification," 1999.
[14] Sun Microsystems, Inc., "Java Secure Socket Extension 1.0.2," 1999.
[15] HAVi, Inc., "Specification of the Home Audio/Video Interoperability (HAVi) ," May 2001.
[16] DAVIC, "DAVIC 1.4.1 Specification Part 9 : Information Representation," 1999.
[17] Edward M. Schwalb, "iTV Handbook," Prentice Hall, 2004.
[18] 신승호, 정문열, "연동형 데이터방송 저작도구의 설계", 방송공학회지, 제9권 제4호, 2004.
[19] 정문열, 백두원, "연동형 데이터 방송 에플리케이션의 구조", 방송공학회논문지, 제9권 제1호, 2004.
[20] 임현정, 임순범, "데이터 방송 미들웨어에서 콘텐츠 호환을 위한 클래스 라이브러리의 설계", 정보처리학회지 제11권 제5호, 2004.
[21] JHotDraw Homepage: http://www.jhotdraw.org


## 저 자 소 개


**이 차 원**

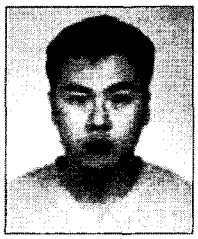
· 2004년 2월 : 한국외국어대학교 컴퓨터공학과 졸업 (학사)
· 2005년 3월 ~ 현재 : 서강대학교 영상대학원 미디어공학과 석사과정
· 주관심분야 : Middleware, Java, iTV Contents



**남 윤 석**

· 2005년 2월 : 서강대학교 화학공학과 졸업 (학사)
· 2005년 3월 ~ 현재 : 서강대학교 영상대학원 미디어공학과 석사과정
· 주관심분야 : New Media, Digital Content, IPTV

## 저 자 소 개

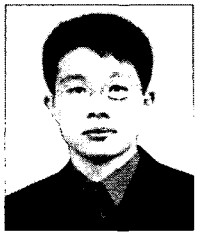



### 김 정 환

- 2004년 2월 : 아주대학교 미디어학부 졸업 (학사)
- 2006년 8월 : 서강대학교 영상대학원 미디어공학과 졸업 (석사)
- 2006년 9월 ~ 현재 : 서강대학교 영상대학원 미디어공학과 박사과정
- 주관심분야 : iTV, IPTV, MPEG-4, 3DTV, 홀로그램

### 나 회 주

- 2005년 2월 : 인하대학교 기계공학과 졸업 (학사)
- 2005년 3월 ~ 현재 : 서강대학교 영상대학원 미디어공학과 석사과정
- 주관심분야 : 양방향 데이터방송 콘텐츠 기술



### 김 성 원

- 2000년 2월 : 대구대학교 경영학과 졸업 (학사)
- 2006년 3월 ~ 현재 : 서강대학교 영상대학원 미디어공학과 석사과정
- 주관심분야 : 양방향 방송 응용 프로토콜

### 정 문 열

- 1980년 2월 : 서울대학교 계산통계학과 졸업 (학사)
- 1982년 2월 : 한국과학기술원 전산학과 졸업 (석사)
- 1992년 2월 : Univ. of Pennsylvania, 전산학과 (박사)
- 1992년 ~ 1994년 : 일본 구주공업대학 조교수
- 1994년 ~ 1999년 : 숭실대학교 컴퓨터학부 부교수
- 현 서강대학교 영상대학원 미디어공학과 정교수
- 주관심분야 : 인터랙티브 방송, 컴퓨터 그래픽스