http://dx.doi.org/10.3745/KTSDE.2013.2.10.669

Design and Implementation of a Protocol for Interworking Open Web Application Store

Jihun Baek[†] · Jihun Kim[††] · Yongwoo Nam[†††] · HyungUk Lee[†††]
Sangwon Park[††††] · Jonghong Jeon[†††††] · Seungyoon Lee[††††††]

ABSTRACT

Recently, because the portable devices became popular, it is easily to see that each person carries more than just one portable device and the use of the smartphone stretches as time goes by. After the smartphone has propagated rapidly, the total usage of the smartphone applications has also increased. But still, each application store has a different platform to develop and to apply an application. The application store is divided into two big markets, the Android and the Apple. So the developers have to develop their application by using these two different platforms. Developing into two different platforms almost makes a double development cost. And for the other platforms, the weakness is, which still have a small market breadth like Bada is not about the cost, but about drawing the proper developers for the given platform application development. The web application is rising up as the solution to solve these problems, reducing the cost and time in developing applications for every platform. For web applications don't need to make a vassal relationship with application markets platform. Which makes it possible for an application to operate properly in every portable devices and reduces the time and cost in developing. Therefore, all of the application markets could be united into one big market through a protocol which will connect each web applications market. But, still there is no standard for the web application store and no current web application store is possible to interlock with other web application stores. In this paper, we are trying to suggest a protocol by developing a prototype and prove that this protocol can supplement the current weakness.

Keywords : Web Application Store, Web Application, Interworking of Open Web Appplication Store

개방형 웹 애플리케이션 스토어 연동을 위한 프로토콜의 설계 및 구현

백 지 훈[†] · 김 지 훈[††] · 남 용 우[†††] · 이 형 욱[†††] · 박 상 원[††††] · 전 종 홍[†††††] · 이 승 윤[††††††]

요 약

최근 휴대용 기기들이 대중화되어 한 사람이 하나 이상의 휴대용 기기를 소지하고 있고 스마트폰 활용도 또한 늘어나는 추세이다. 스마트폰이 많이 보급됨에 따라 폭발적으로 스마트폰의 애플리케이션 활용이 늘어나고 있다. 현재의 애플리케이션 스토어는 플랫폼별로 애플리케이션을 개발해줘야 하는 종속적인 면이 있다. 앱스토어는 크게 애플의 앱스토어와 구글의 안드로이드 마켓으로 양분되어 있고 각 플랫폼에 맞춰 애플리케이션이 개발되어야 한다. 각각의 플랫폼에 맞춰서 애플리케이션을 개발하면 개발 비용은 2배에 근접하고 다른 소규모 플랫폼(Ex 바다)들은 애플리케이션 개발자들을 모아야만 자신들의 플랫폼에 맞는 애플리케이션이 나오는 단점이 있다. 이러한 플랫폼에 맞춰서 개발된 네이티브 애플리케이션의 범용성에 대한 해결책과 모바일에서의 다양한 요구사항을 수용하기 위해 웹 애플리케이션이 각광받고 있다. 웹 애플리케이션은 플랫폼에 종속되지 않고 어느 휴대용 기기에서도 동작하기 때문에 각 플랫폼별로 개발하지 않아도 된다는 장점이 있다. 따라서 웹 애플리케이션 스토어끼리의 연동 프로토콜을 통해 애플리케이션을 연동하여 어느 웹 애플리케이션 스토어에서도 볼 수 있고 특정 플랫폼에 구애받지 않는 거대한 시장이 생겨날 수 있다. 하지만 아직 웹 애플리케이션 스토어라는 표준이 없고 존재하는 웹 애플리케이션 스토어가 없다. 이를 위해 본 논문에서는 연동에 관한 프로토콜을 제안하고 구현을 통해 기존의 애플리케이션 스토어의 단점을 보완할 수 있는 방법을 제시한다.

키워드 : 웹 애플리케이션 스토어, 웹 애플리케이션, 애플리케이션 스토어 연동

1. 서 론

2007년 전 세계적으로 애플의 아이폰 열풍과 함께 아이폰에서 동작이 가능한 애플리케이션을 판매하는 애플의 앱스토어의 성공 이후 수많은 단말기 사업자와 플랫폼 사업자, 통신 사업자들은 앞다투어 독자적인 마켓플레이스 구축에 나서고 있다. 이후 스마트폰에서 사용할 수 있는 애플리케이션의 숫자가 크게 증가하였다. 또한, 애플의 아이폰에 대항하는 구글의 안드로이드폰이 생기며 안드로이드 운영체제

※ 이 논문은 한국전자통신연구원 개방형 웹스토어 연동 기술 개발(1년)에 의하여 연구되었음.
† 정 회 원 : 한국외국어대학교 컴퓨터 및 정보통신공학과 석사
†† 준 회 원 : 한국외국어대학교 정보통신공학과 학사
††† 준 회 원 : 한국외국어대학교 컴퓨터 및 정보통신공학과 학사
†††† 종신회원 : 한국외국어대학교 정보통신공학과 부교수
††††† 종신회원 : 한국전자통신연구원 표준연구센터 서비스융합표준연구팀 선임연구원
†††††† 정 회 원 : 한국전자통신연구원 표준연구센터 서비스융합표준연구팀 팀장
논문접수 : 2013년 3월 13일
수 정 일 : 1차 2013년 4월 29일, 2차 2013년 5월 6일
심사완료 : 2013년 5월 22일
* Corresponding Author : Jihun Baek(lacidjun@hufs.ac.kr)

전용의 안드로이드 마켓도 생겨났다. 또한 전 세계 통신사와 단말기 회사가 모여 구글과 애플의 모바일 시장독점을 견제하기 위해 웹 애플리케이션 기반의 WAC(Wholesale Applications Community)을 결성하였다[1].

기존의 네이티브 애플리케이션 스토어[2](애플 스토어, 안드로이드 마켓)는 운영체제와 단말기에 종속적인 단점 때문에 각 플랫폼별로 애플리케이션을 개발해야 한다. 각 플랫폼별로 애플리케이션을 개발해야 하는 단점을 해결하기 위해 등장한 것이 바로 웹 애플리케이션이다[3].

웹 애플리케이션이 등장하게 된 배경에는 여러 사업자들이 한 플랫폼에 맞춰 개발하기보다는 웹 표준 언어를 사용하여 애플리케이션을 개발하면 플랫폼을 가리지 않고 모든 스마트폰을 지원할 수 있는 것에 있다. 본 논문에서 제안하는 웹 애플리케이션 스토어는 바로 이런 웹 애플리케이션을 지원하는 마켓이다. 웹 애플리케이션 스토어는 운영체제, 플랫폼에 종속적이지 않기 위해 웹 표준언어로 제작하며 크로스 플랫폼[4]을 지원한다.

웹 애플리케이션 스토어는 차세대 웹 기반 언어(HTML5 [5] CSS3, Java Script)로 제작된 애플리케이션들로 이루어진 애플리케이션을 배포하는 스토어이며 현재 Chrome Web Store[6], OpenAppMkt[7]등이 있다. 그러나 이러한 웹 애플리케이션 스토어는 Apple의 AppStore와 Google의 Android Market과 같이 서로 상호보완이 이루어지지 않고 웹 애플리케이션이 등록된 스토어에서만 해당 웹 애플리케이션이 존재하게 되어 사용자에게 있어 선택의 폭이 좁아지게 되거나 번거로움이 생기는 단점을 해결하기 위해 OWS를 제안한다.

웹 애플리케이션 스토어 간의 연동을 하기 위해 제시한 웹 애플리케이션 스토어 형태의 OWS(Open Web Store)는 각각의 OWS 간의 연동을 통하여 글로벌한 애플리케이션 연동이 목적이다. 각 플랫폼에 종속적이지 않고 크게 뒤떨어지지 않는 성능을 가진 웹 애플리케이션이 많아진다면 웹 애플리케이션 스토어는 구글과 애플의 독적점인 애플리케이션 시장구조에 대하여 해결책이 될 수 있을 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 연구되던 관련 연구와 애플리케이션 스토어에 대해 기술하고 3장에서는 본 논문에서 제안하는 웹 애플리케이션 스토어인 OWS의 구조에 대하여 살펴보고 4장에서는 OWS 인터페이스를 기술한다. 5장에서는 연동 시나리오를 기초로 한 OWS의 연동 시나리오 구현에 대하여 기술한다. 마지막으로 6장에서는 향후 연구 방향에 대해 설명하고 결론을 맺는다.

2. 관련 연구

2.1 웹 애플리케이션

아이폰 또는 안드로이드 플랫폼과 같은 고정된 특정 플랫폼에서만 동작하는 애플리케이션을 네이티브 애플리케이션이라고 한다. 이 네이티브 애플리케이션은 특정 플랫폼에서 해당 단말기의 성능을 최대한 사용할 수 있기 때문에 최적화된 애플리케이션을 사용할 수 있다. 반대로 웹 애플리케이션은 네이티브 애플리케이션에 비하여 단말기의 자원을 완전히 사용할 수 없기 때문에 네이티브 애플리케이션에 비하여 상대적으로 최적화되어 있지 못하다. 하지만 웹 애플리케이션의 장점은 웹상에서 애플리케이션이 동작하기 때문에 기존의 네이티브 애플리케이션이 가지고 있는 플랫폼 종속이라는 면에서 자유롭다.

웹 애플리케이션은 차세대 웹 표준 언어인 HTML5를 기반으로 하여 웹 애플리케이션의 동작을 인식하는 자바스크립트와 GUI 부분을 담당하는 CSS3로 구성된다. 따라서 기존의 자바 개발자나 웹 개발자들이 쉽게 애플리케이션을 제작할 수 있다. HTML5는 기존의 HTML4나 XHTML에 비해 많은 발전이 이루어진 언어라서 써드파티 플러그인 없이도 많은 작업을 수행할 수 있게 되어 플랫폼에 종속적이지 않고 웹상에서 동작하는 더 나은 웹 애플리케이션을 제작할 수 있다. 또한, W3C[8]에서 웹 애플리케이션을 위해서 모바일 기기에 대한 내부 자원에 접근할 수 있도록 device API[9]를 만들고 있어 좀 더 단말기에 최적화된 웹 애플리케이션이 제작 가능하다.

웹 애플리케이션의 단점인 느린 속도와 내부 자원을 비효율적으로 접근하는 방법을 개선하기 위해 기존의 웹 애플리케이션을 네이티브 애플리케이션으로 바꾸는 프로그램이 있다. 이러한 방법으로 만들어지는 애플리케이션을 하이브리드 애플리케이션이라고 하며 device API를 통하여 웹 애플리케이션을 감싸서 네이티브 애플리케이션으로 만들어 준다. 따라서 어떤 플랫폼에 대한 네이티브 애플리케이션으로 바꿀 수가 있고 특정 플랫폼의 애플리케이션 스토어에 등록하여 판매가 가능하다. 기본적으로 웹 애플리케이션이기 때문에 화면구성과 같은 부분은 네이티브 애플리케이션과 다르게 서버 측에서 동적으로 업데이트가 가능한 장점이 있다.

2.2 TAS(Telco Application Store)

OMA[10]는 모바일 데이터 서비스의 범세계적 활성화를 위해 기술 규격 개발 및 상호 운용성을 검증하기 위한 포럼이다. TAS는 OMA(Open Mobile Alliance)에서 제안하는 애플리케이션 스토어의 구조이며, 이를 구성하기 위한 다양한 모듈, 모듈 사이의 인터페이스와 그 인터페이스에 해당하는 API로 구성되어 있다. 단일 애플리케이션 스토어의 독자적인 기능만 제안하고 있으며 본 논문에서 제안하는 OWS는 기존 TAS에 연동 모듈을 추가하여 웹 애플리케이션 스토어 하나의 독자적인 기능뿐만 아니라 스토어끼리 연동 모듈을 통하여 다양한 애플리케이션을 보유하는데 의의가 있다. Fig. 1은 TAS의 구조이며 총 6개의 모듈과 모듈 간 6개의 인터페이스로 구성되어 있다. 모듈은 애플리케이션을 판매하는 스토어를 구성하는 주체이며 애플리케이션 구매, 판매, 검수 등 각 모듈은 그에 해당하는 기능을 수행한다. 인터페이스는 TAS의 각 모듈 간의 기능에 맞게 수행하는 API의 모음이다.

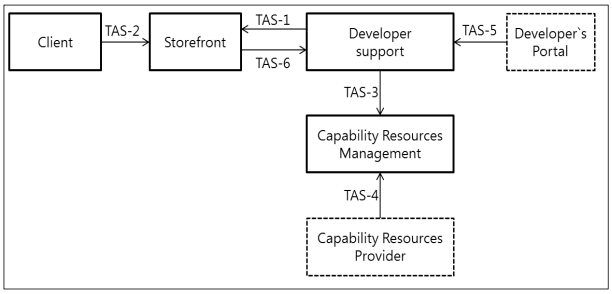


FIg. 1. TAS Structure diagram

2.3 Chrome Web Store

구글의 크롬 웹 스토어는 웹 기반의 앱스토어이다. Installable Web Apps(설치형 웹 앱)라는 형태로 된 패키지 파일을 브라우저로 내려받아 설치하는 방식이다. 크롬 OS는 사용자가 유지, 보수, 관리, 보안을 신경 쓰지 않고 애플리케이션을 편하게 사용만 하면 되는 장점을 지니고 있다. 네트워크 환경만 지원된다면 크롬 북을 이용해 작업했던 데이터들을 다른 곳에서 크롬 북이 아닌 다른 머신을 이용해서 활용하는데 있어서도 매우 유연한 장점을 가지고 있다. 하지만 웹 기반 크롬 브라우저 상태에서 작업을 수행하다 보니 네트워크 환경이 불안정하거나 접속이 끊길 경우 지속적인 작업이 어려운 단점이 있다. 이런 단점을 방지하기 위해 특수한 경우나 몇몇 애플리케이션에서는 오프라인에서 작업을 수행하는 기능을 제공하고 있기도 하다. 또 하나의 단점으로는 크롬 브라우저에서만 애플리케이션을 사용할 수 있기에 크롬 OS에 종속적이다.

2.4 WAC(Wholesale Applications Community)[11]

WAC은 2010년 MWC(Mobile World Congress)에서 창설된 연합체이다. 세계 24개 통신사가 연합하여 새로운 애플리케이션 스토어를 구축하여 Apple과 Google에 집중되어 있는 스마트폰 시장에서 경쟁력을 갖추기 위해 출범했다. 2012년 현재 72개의 통신사 및 제조사가 참여하고 있다.

WAC은 나라별 상위권에 드는 24개 통신사를 바탕으로 하기에 기본적으로 거대한 시장이다. 만약 WAC의 계획대로 애플리케이션 스토어가 만들어진다면 스마트폰 애플리케이션 시장에 엄청난 폭발력을 보일 것이다. 하지만 WAC의 계획대로 진행이 되려면 국제적인 표준 플랫폼 개발이 우선되어야 한다. 또한, 통신사별 이해관계를 잘 해결하지 못하면 수포로 돌아갈 가능성도 크다. 현재 제기되는 통신사 간 이해와 스마트폰의 빠른 변화에 대응하고 시장변화를 따라갈 수만 있다면 WAC의 계획대로 거대한 시장이 생길 수 있을 것이다.

WAC의 플랫폼 구조는 다음과 같다. 개발자들은 WAC에서 제공하는 SDK를 이용하여 웹 애플리케이션을 제작한다. 개발자가 제작한 웹 애플리케이션은 관리자 계정으로 WAC에 등록할 수 있다. 관리자 계정으로 등록한 웹 애플리케이션을 App Store와 연동하면 사용자는 해당 App Store에서 과금 후 다운로드 받아 이용한다.

2.5 K-WAC (Korea-WAC)[12]

K-WAC은 국내 3개 통신사(SKT, LG U+, KT)에 관계없이 애플리케이션을 등록 및 판매를 자유롭게 할 수 있는 한국형 통합 App Store이다. WAC의 한국형 버전으로 전 세계 개발자들이 WAC에 등록한 애플리케이션을 K-WAC 혹은 국내 3대 통신사의 App Store를 통해 판매 가능한 것이 장점이다.[13][14] K-WAC은 WAC의 최초 서드 파티 애플리케이션 마켓으로 WAC에서 애플리케이션 정보를 얻어온다. 그리고 이 정보를 토대로 국내 3사 통신사에 동일한 애플리케이션을 공급할 수가 있다. 이는 애플리케이션 개발자로 하여금 다양한 마켓에서 애플리케이션을 판매할 수 있기 때문에 개발자에게 유리한 환경을 제공한다. Fig. 2는 WAC과 K-WAC 간의 연동과 국내 3사 통신사가 국내외 개발자들이 제작하여 등록한 웹 애플리케이션에 대한 유통 과정을 보여주고 있다.

Fig. 2. Platform structure of K-WAC[15]

2.6 애플리케이션 스토어 분석

애플리케이션 간 비교 분석은 WIP[16](Wireless Industry Partnership)의 애플리케이션 스토어 목록을 참고하여 조사한 후 작성하였다. 또한 추가적으로 인터넷 검색을 통하여 앱 스토어를 검색하였고 구동OS, 연회비, 수익 분배 등의 관련 정보를 제공한다.

애플리케이션 스토어에서 다루는 스마트폰 운영체제의 경우 가장 큰 비율을 차지하는 것이 구글사의 안드로이드OS를 다루는 애플리케이션 스토어 이고[17] 두 번째 로는 애플사의 iOS를 다루는 애플리케이션 스토어가 많았다. Fig. 3은 애플리케이션 스토어에서 다루는 OS에 대한 그래프이다. 하

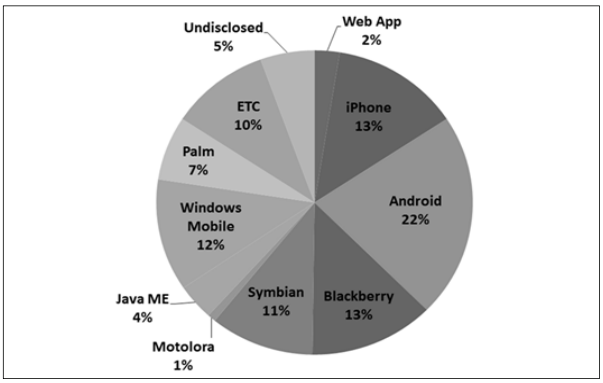


Fig. 3. Operating system to handle the app store

Table 1. Application Store Analysis

	Web App	iPhone	Android	Blackberry	Symbian	Motorola	Java ME	Windows	Palm	ETC	Undisclosed
Single Store	6	14	21	4	3	2	0	6	3	9	13
Complex Store	0	17	30	26	22	0	9	22	13	13	0
Cross Platform	4										
Broweser Dependent	1										
sum	6	31	51	30	25	2	9	28	16	24	13

지만 애플리케이션 전체 시장은 구글사와 애플사의 점유율이 70%이상을 육박한다. 이러한 단점으로 독점이 아닌 크로스 플랫폼을 지원하는 애플리케이션 스토어가 등장하지만 그 영향력을 발휘하기에는 역부족이다. 아래의 Table 1은 WIP에서 제공하는 애플리케이션 스토어 목록 중 스토어를 종류별로 분류하고 수를 나타낸 표이다.

3. OWS 구조

기존의 애플리케이션 스토어의 단점을 극복하기 위한 대안으로 본 논문에서는 OWS(Open Web Store)를 제안한다. OWS는 기존의 애플리케이션 스토어의 단점인 플랫폼에 종속적인 단점을 보완한다. 또한 사용자로 하여금 다양한 애플리케이션을 선택할 수 있게 하며 구글사와 애플사의 애플리케이션 시장 독점 현상을 대처할 수 있는 대안으로 제안한다. 현재 그 요구사항을 국내 표준화 기구(TTA)에 제출한 상태이다.

애플리케이션 개발자가 애플리케이션을 판매할 경우 해당 단말기 운영체제에 맞는 애플리케이션 스토어에만 애플리케이션을 등록할 수 있다. 하지만 웹 애플리케이션 스토어는 표준 웹 언어로 애플리케이션을 제작하기 때문에 플랫폼에 구애받지 않고 다양한 애플리케이션 스토어에 업로드 할 수 있다. 또 애플리케이션 스토어 간 연동을 통해 다양한 애플리케이션을 유통할 수 있는 장점을 가지고 있다.

OWS는 OMA에서 제안한 TAS를 토대로 설계한 애플리케이션 스토어 간 연동이 가능한 개방형 웹 애플리케이션 스토어다. Fig. 4에 기술한 OWS의 구조는 TAS의 Storefront 모듈에 연동 부분을 설계하여 추가함으로써 Extended Storefront 모듈을 설계한 것이다. 기존의 Storefront 모듈과 다른 점은 연동 모듈인 SAM(Shared App Management)모듈과 정책 모듈인 Policy 모듈, Policy Provider 모듈이 추가되어 있다. 또한 각 모듈들의 기능에 맞는 API들의 모음인 인터페이스도 각각 추가되었다. 우선 각 모듈들에 대하여 설명하고 전체적인 구조를 기술하겠다.

3.1 Policy

Policy 모듈은 크게 2가지 기능을 수행한다. 애플리케이션을 송신할 때와 애플리케이션을 수신할 때로 나뉜다. 애플리케이션을 송신할 때는 Policy 모듈이 애플리케이션의 상태를 확인한다. 처음 개발자가 애플리케이션을 등록할 때

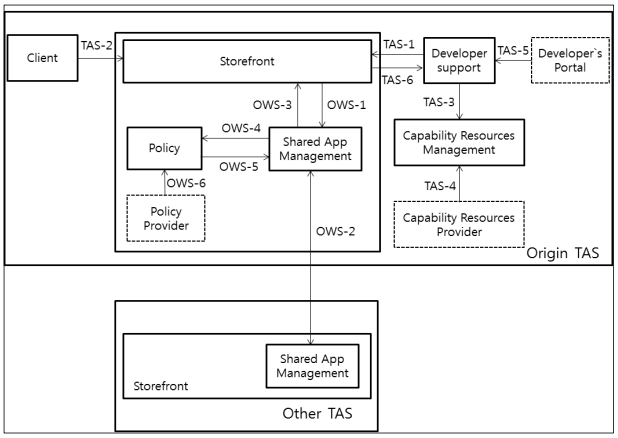


Fig. 4. Structure diagram of OWS

검수 받는 과정과 유사하다. 애플리케이션의 정보 상태, 버전, 데이터, 용량 등이 초기에 Storefront 모듈에 등록되었을 때나 업데이트가 되었을 때의 최근 정보와 같은지 판단하고 다르다면 애플리케이션 전송을 거부하고, 검수가 통과된다면 수신 측 TAS에게 애플리케이션을 전송해도 된다는 결과를 SAM 모듈에 통보한다. 애플리케이션을 수신할 때는 애플리케이션을 송신 측 TAS에서 수신 측 SAM로 전송하면, 전송받은 애플리케이션에 대한 정보가 Policy Provider 모듈로부터 적용된 검수 정책과 수신 측 TAS의 조건에 부합하는지 확인하는 기능을 수행한다.

3.2 Policy Provider

Policy Provider 모듈은 External 모듈로 직접적으로 OWS에서 관여하지 않는다. 각 TAS별로 애플리케이션 검수 기준과 이해관계가 다르기 때문에 그 기준에 부합하는 정책을 각 TAS마다 다르게 부여한다. 외부 기관에서 TAS에 애플리케이션 검수하는데 필요한 조건을 보내고 TAS는 이 검수 정책에 맞게 애플리케이션 및 정보를 검수하는 기능을 수행한다.

3.3 SAM(Shared App Management)

SAM 모듈은 TAS내에서 실질적인 애플리케이션 연동을 위한 모듈이다. 애플리케이션 연동에 관한 모든 일들은 SAM 모듈을 거치게 된다. 한 웹 스토어에서 애플리케이션을 수신 측 TAS로 보내기 위해 최초로 수신 측 Store front 모듈에서 애플리케이션이나 리스트를 받아 Policy 모

듈의 검수과정을 거친 후 송신 측 TAS의 SAM로 전송하는 과정을 수행하고, 애플리케이션을 수신할 때에도 Policy 모듈의 검수과정을 거친 뒤 Storefront 모듈로 애플리케이션을 전송하는 기능을 수행한다.

3.4 OWS 인터페이스

본 논문에서 제안하는 SAM 모듈, Policy Provider 모듈, Policy 모듈 간의 인터페이스는 총 6가지로 구성된다. 제안하는 6가지 인터페이스에는 그 기능을 수행하기 위한 API로 구성된다. 다음의 Table 2는 모듈 간 6가지의 인터페이스의 기능에 대한 표이다.

Table 2. OWS interface description

Interface	Descriptions
OWS-1	The interface that the application and the application list are sent from the storefront to the receiver, the SAM module
OWS-2	The interface that information about applications communicate between origin SAM and other SAM
OWS-3	The interface that checked applications and received applications are sent to storefront for registration
OWS-4	The interface that checking applications received from receiver side of OWS
OWS-5	The interface that reporting the application which not pass the checking
OWS-6	The interface that is to decide policy and check contents for

3.5 모듈 간 연동 송신, 수신 순서도

Fig. 5는 OWS간 애플리케이션을 공유하기 위해 송신 측에서 수행되는 인터페이스 흐름을 보여주는 순서도 이다. 수신 측 OWS의 Storefront 모듈로 애플리케이션을 연동하기 위한 모듈인 SAM 모듈로 애플리케이션의 리스트 및 관련 정보들을 전송한다. 이 후 애플리케이션의 이상 유무 확인을 위해 Policy 모듈에서 검수확인을 거친 뒤 연동을 하기 위한 수신 측 OWS의 SAM 모듈로 애플리케이션 리스트 및 관련 정보들을 전송한다.

Fig. 6은 OWS간 애플리케이션을 공유하기 위해 수신 측에서 수행되는 인터페이스 흐름을 보여주는 순서도 이다. 송신 측에서 전송 받은 애플리케이션 관련 정보를 Storefront 모듈로 전송하기 전 해당하는 국가나 지역 혹은 OWS에 정책, 이해관계에 의해서 제한되는 내용에 대해 Policy 모듈에서 검수작업을 거치게 된다. 검수작업을 마친 애플리케이션은 최종적으로 다른 OWS에 등록이 되기 위해 Storefront 모듈로 애플리케이션을 전송하게 된다. 연동이 되는 마지막 과정은 애플리케이션 연동을 요청해온 OWS에 결과를 전송한다.

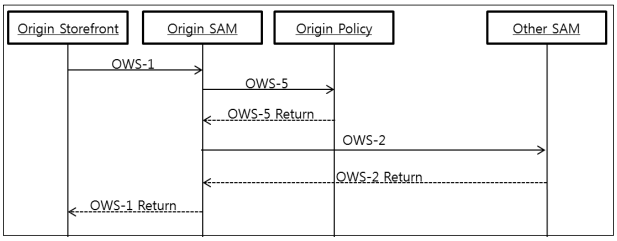


Fig. 5. Interface sequence diagram between the sender module

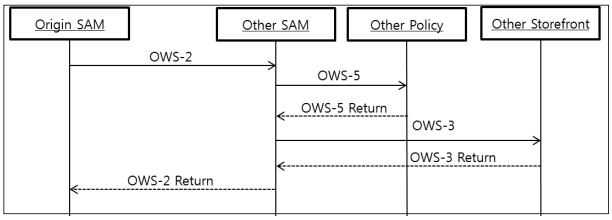


Fig. 6. Interface sequence diagram between the receiver module

4. OWS 인터페이스의 API 리스트

OWS의 API 리스트는 각 OWS 인터페이스마다 존재한다. OWS의 인터페이스에 있는 API는 연동하기 위해 필요한 기능 부분을 설계 하였다. 현재 웹 애플리케이션 연동 프로토콜 요구사항은 국내 표준화 기구(TTA)에 제안한 상태이다. 각 OWS 인터페이스별로 API 리스트를 설명한다.

4.1 OWS-1

애플리케이션을 연동하기 위한 모듈인 SAM 모듈에서 Storefront 모듈에게 애플리케이션 리스트를 요청하거나 수신 측 Storefront 모듈에서 애플리케이션 연동 요청이 들어왔을 때 애플리케이션을 연동하는 기능, 또 다른 OWS에게서 받아온 애플리케이션이 송신 측 애플리케이션을 등록 한 개발자나 불량 신고로 인해 삭제되었을 때 애플리케이션이 연동 된 수신 측 애플리케이션도 삭제하기 위한 삭제 혹은 업데이트 연동 기능, 사용자의 구매, 환불에 의한 정보 연동 등에 대한 기능이 이루어지는 인터페이스다. 다음 Table 3은 OWS-1 인터페이스를 구성하는 API 리스트이다.

Table 3. OWS-1 interface API list

API Name	Descriptions
Shared App Deletion	Deleting an application
Shared App Update	Updating an application
Sender OWS App List	Request the list of applications
Receiver OWS App Share	Request application interworking
Shared App Payment	Application billing
Shared App Refund	Application refunding

4.2 OWS-2

OWS-2 인터페이스는 연동하기 위해 송신 측 OWS의 SAM 모듈과 수신 측 OWS의 SAM 모듈 간의 인터페이스이다. 송신 측에서는 애플리케이션 리스트 혹은 애플리케이션 정보들을 Storefront에서 전송 받은 뒤 수신 측 SAM으로 정보를 전달하는 기능을 수행한다. 다음 Table 4는 OWS-2 인터페이스를 구성하는 API 리스트이다.

Table 4. OWS-2 interface API list

API Name	Descriptions
App Information Transfer	Register application
App Deletion Transfer	Transfer deleting information of application
App Download Request	Request application download
App Feedback Transfer	Application feedback
App Total Feedback Information	Total feedback of application
Malicious And Bug Report	Application report
App Charge Delivery	Transfer billing information of application
App List Transfer Request	Transfer application list

4.3 OWS-3

OWS-3 인터페이스는 OWS-4와 OWS-5 인터페이스를 통해 검수받은 애플리케이션 및 인접 OWS에게 전송받은 애플리케이션을 등록하기 위해 Storefront에 전송하는데 사용되는 인터페이스이다. 다음 Table 5는 OWS-3 인터페이스를 구성하는 API 리스트이다.

4.4 OWS-4

OWS-4는 수신 측 OWS에서 전송받은 애플리케이션을 검수하는데 사용되는 인터페이스이다. 송, 수신하는 관점에서 두 가지의 기능을 수행한다. 애플리케이션 정보들을 전송할 때는 Storefront에 등록된 정보와 부합하고 변화가 있는지 등에 대한 정보를 검수하고 수신 측에서의 OWS-4 인터페이스는 수신 측 조건에 맞는지 검수하는 기능을 수행한다. 다음 Table 6은 OWS-4 인터페이스를 구성하는 API리스트이다.

Table 5. OWS-3 interface API list

API Name	Descriptions
Shared App List	Transfer application list for sharing
Shared App Transfer	Transfer application
Malicious Shared App Report	Transfer application report
Shared App Refund	Refund shared application
Last App Pay Request	Transfer commission of application

Table 6. OWS-4 interface API list

API Name	Descriptions
Application Check	Application inspection

4.5 OWS-5

OWS-5 인터페이스는 OWS-4인터페이스를 통한 애플리케이션 검수가 통과되지 않은 경우 지속적으로 해당 애플리케이션에 대한 보고서를 보내는데 사용되는 인터페이스이다. 다음 Table 7은 OWS-5 인터페이스를 구성하는 API 리스트이다.

Table 7. OWS-5 interface API list

API 이름	Descriptions
Application Policy Report	Transfer the results of application transferred

4.6 OWS-6

OWS-6 인터페이스는 SAM에서 전송받은 애플리케이션을 검수하기 위한 내용 및 정책 등을 결정하기 위한 인터페이스이다. OWS는 지역에 따라 다른 이해관계나 정책 등을 결정하는 외부 모듈(Policy Provider)에서 Policy 모듈로 제공하는 기능을 수행한다. 외부 모듈이기 때문에 API가 존재하지 않으며 지역에 따른 검수 정책을 Policy 모듈에 공급하는 기능을 수행한다.

5. 연동 시나리오 및 구현

Fig. 7은 애플리케이션 송신 측 Storefront 모듈과 수신 측 Storefront 모듈 간 애플리케이션 리스트 연동 후 리스트 목록에서 연동하길 원하는 애플리케이션을 요청하는 순서도이다.

5.1 연동 시나리오

연동 시나리오는 애플리케이션이 등록되어 있는 송신 측 Storefront 모듈은 데이터베이스에 애플리케이션 정보를 가

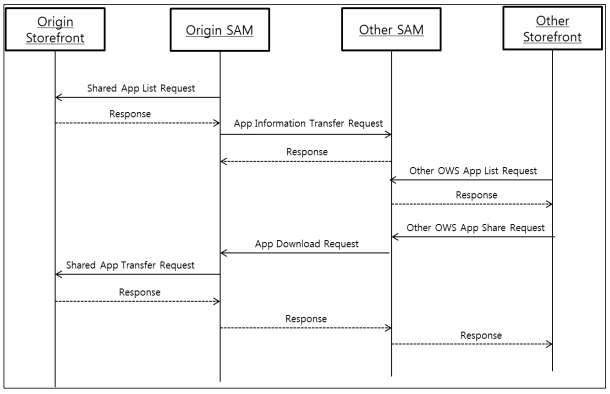


Fig. 7. Application interworking sequence diagram

지고 있다고 가정한다. SAM 모듈은 Storefront 모듈에 애플리케이션 리스트를 요청하고, 수신 측 Store의 SAM 모듈로 애플리케이션 리스트를 전송한다. 수신 측 SAM 모듈은 전송받은 애플리케이션 리스트를 수신 측 Storefront 모듈에 전송하며, 수신 측 Storefront 모듈은 송신 측 Storefront모듈 에서 전송받은 애플리케이션 리스트에 있는 애플리케이션 중 연동을 원하는 애플리케이션의 ID로 애플리케이션 요청을 하게 되고, 송신 측 Storefront는 서버의 데이터베이스에 저장된 해당 애플리케이션을 수신 Storefront 모듈로 전송하는 시나리오를 설계하였다.

5.2 애플리케이션 연동 시나리오 구현

본 논문에서 제안하는 개방형 웹 애플리케이션 스토어의 아키텍처, 모듈, 인터페이스, API 를 토대로 구현하였다. 서버는 구글 앱 엔진[18]을 사용하였으며 애플리케이션 연동을 위한 애플리케이션 스토어는 총 3개로 구현하였다. 애플리케이션을 가지고 있는 First Store와 애플리케이션이 연동되는 Second Store, Third Store로 애플리케이션 연동 테스트를 수행한다.

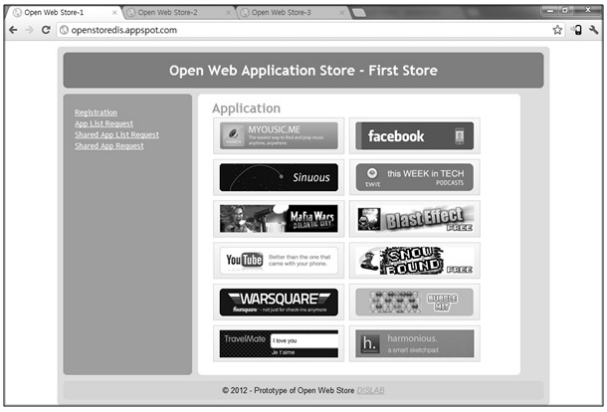


Fig. 8. Main of First Store

총 3개의 애플리케이션 스토어 중 애플리케이션 목록을 모두 가지고 있는 스토어는 Fig. 8의 First Store이다. Fig. 9와 Fig. 10은 First Store에서 애플리케이션을 연동할 수신 측 애플리케이션 스토어 이다.

First Store는 총 12개의 애플리케이션을 가지고 있으며, Second Store와 Third Store로 애플리케이션 리스트와 애플리케이션을 연동하는 역할을 수행한다. Second Store와 Third Store는 First Store에서 애플리케이션을 연동 받아 각각 5개, 1개의 애플리케이션을 저장하고 있다.

애플리케이션을 연동하는 처음 과정은 First Store의 SAM 모듈에서 First Store Storefront에 데이터베이스에 생성, 저장해놓은 애플리케이션 리스트를 요청하는 것이다. 데이터베이스에 저장해놓은 애플리케이션 리스트는 XML[19] 파일 구조로 저장되어 있다.

Fig. 8의 왼쪽 메뉴 중 App List Request를 누르면 SAM 모듈에서 Storefront 모듈로 애플리케이션 리스트를 요청한

Fig. 9. Main of Third Store

Fig. 10. Main of Second Store

다. App List Request를 누르면 Table 5에서 OWS-3 인터페이스에 정의되어 있는 Shared App List API를 이용하여 SAM 모듈에서 Storefront 모듈로 애플리케이션 리스트를 요청하고, Fig. 5에 송신측 모듈 간 연동 순서도에서 OWS-1 인터페이스를 통해 요청된 애플리케이션 리스트를 XML 파일 형식으로 SAM 모듈로 전송한다. Fig. 11은 요청 완료 후 First Store Storefront에서 First Store SAM 모듈로 전송되는 XML 파일이다.

First Store는 총 12개의 애플리케이션을 가지고 있으므로 Fig. 11과 같이 12개의 애플리케이션 정보가 담긴 리스트가 전송된다. 현재 애플리케이션 리스트는 First Store의 SAM 모듈에 존재하고 있으므로 Second Store와 Third Store의 SAM 모듈로부터 애플리케이션 리스트 요청이 오면 Fig. 11의 XML 파일을 전송한다.

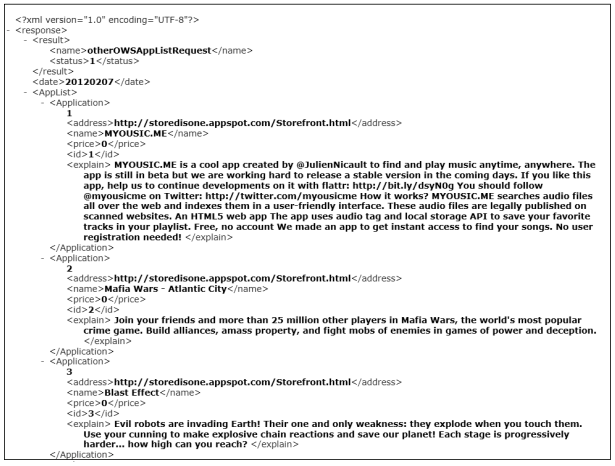


Fig. 11. XML file that was sent from first store SAM module to second storefront module

<?xml version="1.0"?>
- <response>
  - <result>
    <name>SharedAppListRequest</name>
    <status>1</status>
  </result>
  - <AppList>
    - <Application>
      1
      <address>http://storedisone.appspot.com/Storefront.html</address>
      <name>MYOUSIC.ME</name>
      <price>0</price>
      <id>1</id>
      <explain> MYOUSIC.ME is a cool app created by @JulienNicault to find and play music anytime, anywhere. The app is still in beta but we are working hard to release a stable version in the coming days. If you like this app, help us to continue developments on it with flattr: http://bit.ly/dsyN0g You should follow @myousicme on Twitter: http://twitter.com/myousicme How it works? MYOUSIC.ME searches audio files all over the web and indexes them in a user-friendly interface. These audio files are legally published on scanned websites. An HTML5 web app The app uses audio tag and local storage API to save your favorite tracks in your playlist. Free, no account We made an app to get instant access to find your songs. No user registration needed! </explain>
    </Application>
    - <Application>
      2
      <address>http://storedisone.appspot.com/Storefront.html</address>
      <name>Mafia Wars - Atlantic City</name>
      <price>0</price>
      <id>2</id>
      <explain> Join your friends and more than 25 million other players in Mafia Wars, the world's most popular crime game. Build alliances, amass property, and fight mobs of enemies in games of power and deception. </explain>
    </Application>
    - <Application>
      3
      <address>http://storedisone.appspot.com/Storefront.html</address>
      <name>Blast Effect</name>
      <price>0</price>
      <id>3</id>
      <explain> Evil robots are invading Earth! Their one and only weakness: they explode when you touch them. Use your cunning to make explosive chain reactions and save our planet! Each stage is progressively harder... how high can you reach? </explain>
    </Application>

Fig. 12. XML file that was sent from first store storefront module to first store SAM module

OWS에서 애플리케이션 연동을 하기 위해서는 SAM 모듈 간의 연동 과정이 필요하다. First Store SAM 모듈에 저장되어 있는 애플리케이션 리스트를 Second Store SAM 모듈과 Third Store SAM 모듈로 연동하기 위해서는 Second Store SAM 모듈 혹은 Third Store SAM 모듈에서 First Store SAM 모듈로 애플리케이션 리스트를 요청해야 한다. 다음 과정은 SAM 모듈 간 애플리케이션 리스트 연동하는 과정이다.

Fig. 10의 아래 그림 왼쪽 메뉴 중 Shared App List Request를 누르면 Table 4의 OWS-2 인터페이스의 App Information Transfer API를 통해 송신 측 SAM 모듈과 수신 측 SAM 모듈간의 애플리케이션 리스트 연동이 이루어진다. 이 과정은 Fig. 5와 Fig. 6에서 OWS-2 에 해당하는 부분이다. 그 결과 Fig. 12의 XML 파일이 Second Store SAM 모듈로 전송되는 것을 확인할 수 있다.

현재 애플리케이션 리스트는 Second Store 혹은 Third Store 의 SAM 모듈에 존재한다. 애플리케이션 리스트를 Table 5와 같이 OWS-3 인터페이스의 수신 측 OWS App List API를 통해 수신 측 SAM 모듈에 존재하는 송신 측 애플리케이션 리스트를 Storefront에 전송한다. 이 부분은 Fig. 6의 순서도에서 수신 측 SAM 모듈과 수신 측 Storefront에서 OWS-3 인터페이스로 정보를 연동하는 부분이다.

애플리케이션 연동 마지막 과정은 수신 측 Storefront 모듈에서 애플리케이션 리스트를 확인한 후 연동을 원하는 애플리케이션 ID를 해당 애플리케이션이 등록되어 있는 웹 애플리케이션 주소로 요청하면 해당 ID의 애플리케이션 정보가 담긴 XML 파일이 First Store에서 Second Store 혹은 Third Store로 전송된다. Fig. 8의 Facebook 애플리케이션 연동을 원할 때 해당 애플리케이션 ID(XML 파일에 있는 Facebook 애플리케이션의 ID는 10이다.)를 First Store의 주소로 애플리케이션 연동을 요청한다. 이 때 사용되는 API는 Fig. 7의 OWS-1 인터페이스인 수신 측 OWS App Share를 사용하여 SAM 모듈에게 애플리케이션 연동을 요청한다.

<?xml version="1.0" encoding="UTF-8"?>
- <application>
  <img>http://postfiles12.naver.net/20120221_203/knowkjh
  <name>Facebook</name>
  <id>10</id>
  <price>0</price>
  <address>http://touch.facebook.com/?app_id=4c3eac62c441
  <explain>Facebook is a social utility that connects you to the messages</explain>
</application>

Fig. 13. Facebook application XML file

Fig. 14. Facebook application of Second store

Fig. 15. Facebook application of Third store

Second Store SAM 모듈과 Third Store SAM 모듈은 Facebook 애플리케이션을 연동하기 위해 First Store에 Fig. 7의 OWS-2 인터페이스의 App Download Request를 통해 애플리케이션 연동 요청을 한다. 연동 요청을 받은 First Store SAM모듈은 연동 요청 정보를 Fig. 7에서 보이듯이 OWS-3 인터페이스의 Shared App Transfer Request API를 통해 애플리케이션 연동 요청을 하고 애플리케이션은 해당 API의 Response를 통해 Facebook 애플리케이션 정보를 전달한다.

Facebook 애플리케이션 정보는 Fig. 13과 같이 XML파일 형식으로 Response API를 통해 Seconds Store 혹은 Third Store로 전송된다.

XML은 파싱되어 수신 측 Storefront 모듈의 데이터베이스에 저장되고 애플리케이션의 연동이 완료된다.

Fig. 14와 Fig. 15는 Facebook 애플리케이션이 연동 된 후의 Storefront 모습이다. Fig. 9와 Fig. 10과 비교해 볼 때 Facebook 애플리케이션이 생긴 것을 확인 할 수 있다.

6. 결론 및 향후 연구 과제

웹 애플리케이션으로 인해 애플의 iOS, 구글의 Android 등 각 플랫폼에 종속되지 않는 크로스 플랫폼이 각광받고

있다. 하지만 아직 웹 애플리케이션에 대한 표준과 웹 애플리케이션 스토어에 대한 표준이 나와 있지 않다. 그래서 아직 시장은 형성되지 않아있고 그렇기에 본 논문에는 웹 애플리케이션에 맞는 웹 애플리케이션 스토어에 대한 표준 연구를 진행하였다.

본 논문에서는 TAS 표준을 토대로 하여 가상의 웹 애플리케이션 스토어를 만들고 이에 대한 요구사항에 맞춘 웹 스토어끼리의 연동을 보이고 있다. 이를 보이기 위해 웹 애플리케이션에 대한 전반적인 이해와 구조설계를 위하여 국내외 웹 애플리케이션 스토어에 대한 조사를 진행하였다. 또한 웹 애플리케이션 스토어의 전체 흐름을 이해하기 위해 개발자 입장에서의 연동 시나리오와 소비자 입장에서의 연동 시나리오, 웹 스토어끼리 연동을 위한 시나리오를 제시한다. 또한 각 입장에서의 필요한 요구사항들에 맞춘 웹스토어 연동에 필요한 모듈들을 제안한다.

본 논문은 요구사항을 토대로 OWS의 연동 모듈들을 설계하였다. 연동 모듈은 기존 OMA의 TAS 아키텍처를 착안하여 TAS 구조 중 Storefront 모듈 부분에 SAM 모듈과 Policy 모듈, Policy Provider 모듈을 추가하여 확장 설계하였다. 이 연동 모듈들을 통하여 OWS간의 연동을 가능하게 함으로써 웹 애플리케이션이 한 스토어에 종속되지 않고 스토어끼리 연동을 통해 소비자로 하여금 다양한 웹 애플리케이션을 접하게 할 수 있다.

본 논문에서 구상한 시나리오에 입각한 구현은 서블릿 &JSP, Google App Engine 기술을 사용하여 가상의 OWS를 만들었고 이 OWS를 통해 웹 애플리케이션의 연동 시나리오를 보였다.

향후 연구로는, 본 논문에서 구현한 시나리오 부분에서 더 나아가 개방형 웹 애플리케이션 스토어의 과금, 검수, feedback등에 대한 다양한 연동 부분에 대해 연구가 진행되어야 할 것이다. 또한 현재 OWS는 URL 형식의 웹 애플리케이션만을 다루고 있기 때문에 향후 설치형 웹 애플리케이션 연동에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] 이승윤, 정해원, “모바일 웹 2.0 표준화 동향 및 전장”, 한국정보처리학회, Vol.15, No.4, 2008.

[2] 이승윤, “앱스토어 표준화 전략”, TTA, TTA Journal No.131, 2010.

[3] 이승윤, 권성인, “차세대 모바일 웹, 모바일 OK 표준화 전략”, TTA, TTA Journal No.132, 2010.

[4] 전종홍, 이승윤, “차세대 모바일 웹 애플리케이션 표준화 동향”, ETRI, 전자통신동향 분석 제 25권 1호, 2010.

[5] W3C, Working Draft, “HTML5”, http://www.w3.org/TR/html5

[6] Chorme Web Store, http://chrome.google.com/webstore

[7] OpenAppMkt, http://openappmkt.com/

[8] W3C MobileOK, http://www.w3.org/TR/mobileOK-basic10-tests/

[9] W3C Device API WG, http://www.w3.org/2009/dap/

[10] OMA, http://www.openmobilealliance.org

[11] WAC, http://www.wacapps.net/

[12] K-WAC, http://www.wacapps.net/

[13] 이상산, “한국 통합 앱 스토어 추진현황 및 향후 계획”, http://www.moiba.org/, 2011.

[14] 윤상원, “Current & Future of WAC/KWAC", INFRAWARE 모바일사업본부, 2011.

[15] WISEWIRES, “Korea WAC Introduction plan and standard-Korea WAC”, MOIBA, WAC & K-WAC, 2011.

[16] WIP, “Application Store Report”, http://www.wipconnector.com, 2011.

[17] 이선영, “구글의 모바일 비즈니스 추진 동향”, KISDI 방송통신정책, 제 21권, 제 15호, 통권 468호.

[18] google App Engine : http://appengine.google.com/

[19] Extensible Markup Language(XML) 1.0, W3C Recommendation.

백 지 훈

e-mail : lacidjun@hufs.ac.kr

2004년 한국외국어대학교 정보통신공학부 (학사)

2011년~현 재 한국외국어대학교 컴퓨터 및 정보통신공학과 석사

관심분야 : 모바일 컴퓨팅, 데이터베이스, 플래시 메모리

김 지 훈

e-mail : knowkjh@hufs.ac.kr

2005년 한국외국어대학교 정보통신공학과 (학사)

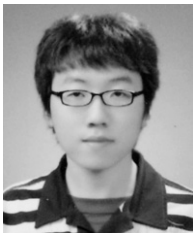
관심분야 : 모바일 컴퓨팅, 데이터베이스, 플래시 메모리

남 용 우

e-mail : skadyddn@hufs.ac.kr

2008년~현 재 한국외국어대학교 컴퓨터 및 정보통신공학과 학사

관심분야 : 모바일 컴퓨팅, 데이터베이스, 플래시 메모리



이 형 욱

e-mail : lhu114@hufs.ac.kr

2008년~현 재 한국외국어대학교 컴퓨터 및 정보통신공학과 학사

관심분야 : 모바일 컴퓨팅, 데이터베이스, 플래시 메모리

박 상 원

e-mail : swpark@hufs.ac.kr
1994년 서울대학교 컴퓨터공학과(학사)
1997년 서울대학교 컴퓨터공학과(석사)
2002년 서울대학교 컴퓨터공학과(박사)
2002년～2003년 세종사이버대학교 디지털 콘텐츠학과 전임강사
2010년～현 재 한국외국어대학교 정보통신공학과 부교수
관심분야 : 모바일 컴퓨팅, 데이터베이스, 플래시 메모리

전 종 홍

e-mail : hollobit@etri.re.kr
1996년～1999년 한국정보시스템 기술개발 연구소 주임연구원
1999년～현 재 TTA 웹프로젝트 그룹 (PG605) 부의장
2008년～현 재 TTA 모바일 웹 실무반 (WG6051) 의장
2009년～현 재 모바일 웹 2.0 포럼 One Widget AG 의장
2006년～현 재 TTA 국제표준전문가
현 재 한국전자통신연구원 표준연구센터 서비스융합표준연구팀 선임연구원
관심분야 : 모바일 웹, 웹 2.0 응용, 유비쿼터스 웹, 소셜 웹, 웹 기술 표준화

이 승 윤

e-mail : syl@etri.re.kr
1999년 ETRI 표준연구센터 입사 (선임연구원)
2003～현 재 ETRI 표준연구센터 서비스융합표준연구 팀장(책임연구원)
2004년～현 재 TTA 국제표준전문가
2006년～현 재 TTA 웹프로젝트 그룹 (PG605) 의장
2005년～현 재 ASTAP IRT EG 라포쳐
2006년～현 재 ITU-T SG13 Editor
2008년～현 재 W3C 대한민국사무국 사무국장
2009년～현 재 ISO/IEC JTC 1 SC38 SGCC 컨비너
현 재 한국전자통신연구원 표준연구센터 서비스융합표준연구팀 팀장
관심분야 : 차세대 웹, 모바일 웹, 유비쿼터스웹, 클라우드컴퓨팅, IPTV, 미래인터넷, e-Book 표준 등