모바일 서비스 아키텍처(MSA) 표준화 동향

커넥티드시스템즈 강성천

1. 서 론

MSA(Mobile Service Architecture)는 노키아와 보다폰의 주도로 만들어진 차세대 자바 모바일 플랫폼 규격으로 이동 통신 단말기에 탑재 되어 실행되는 각종 어플리케이션을 배포하고 실행 시키기 위한 자바 기반의 개방형 표준 플랫폼을 정의한다.

MSA의 표준화 작업은 JCP(Java Community Process) 내에서 JSR(Java Specification Request) 규격으로 진행되고 있으며 JSR-248(Mobile Service Architecture) 규격과 JSR-249(Mobile Service Architecture Advanced) 규격으로 구성된다.

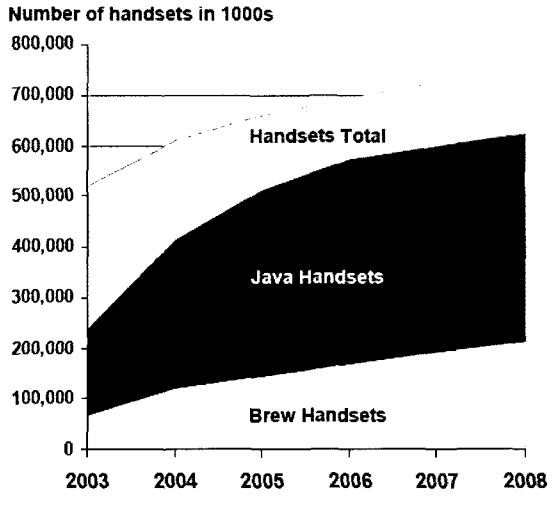

그림 1 세계 핸드셋 자바 탑재 현황[1)]

각각의 JSR 규격은 2004년 8월에 승인되었으며 현재 JSR-248 규격은 Proposed Final Draft 상태이고 JSR-249 규격은 Expert Group Foundation 상태이다. JSR-248 규격은 2006년 8월경에 최종 규격이 완료 될 것으로 예상되며 JSR-249 규격은 2006년도 4분기 경에 Early Draft Review가 공개될 것으로 예상된다.

MSA는 플랫폼을 정의하기 위한 별도의 새로운 표준 규격을 제정하기 보다는 JCP의 다른 JSR 규격 및 OMA(Open Mobile Alliance), OMTP(Open Mobile Terminal Platform), OSGi(Open Service Gateway initiative) 등과 협력하여 표준화를 진행하고 있다.

MSA는 개방형 표준들을 폭넓게 수용할 뿐만 아니라 썬, 모토롤라, 오렌지 모바일, T-모바일, 지멘스, 소니 에릭슨 등의 이동 통신 시장의 주요 기업들로부터 폭 넓은 지지를 받고 있고 그림 1과 같이 자바는 이미 이동 통신 시장에서 폭넓게 채택 되어 있기 때문에 2006년 하반기부터 시장에서 빠르게 채택될 것으로 예상된다. 이미 노키아에서는 올해 JavaOne 컨퍼런스에서 MSA에 기반을 둔 이동 통신 단말을 시연하였다.

2. MSA의 등장 배경

MSA의 표준화 배경에는 JSR-185(JTWI : Java Technology for the Wireless Industry) 규격이 존재한다.

이동 통신사들이 Java 2 ME(Micro Edition) CLDC (Connected Limited Device Configuration) 및 MIDP(Mobile Information Device Profile) 규격을 도입하면서 차별화를 위해 독자적인 API 규격과 프로파일을 확장하였고 이로 인해 비표준 플랫폼이 난립하게 되어 업계로부터 호환성에 관한 문제가 제기되었다. JSR-185는 이와 같은 호환성 문제를 최소화하기 위하여 썬의 주도로 2003년에 제정되었다.

그러나 JSR-185 규격은 CLDC/MIDP 규격에 기반 한 최소한의 요구 사항과 JSR 규격만을 규정하고 있기 때문에 호환성을 유지하면서 최신 기술을 수용하고 풍부한 모바일 어플리케이션을 제공하고자하는 이

1) Asko Komsi, Mark Düsener, "Mobile Service Architecture Initiative: The Latest News on JSR 248 and 249" 2005 JavaOne Conference, 2005.5

동 통신 시장의 요구를 만족시키기에 많은 어려움이 있다.

이와 같은 시장 및 업계 요구에 의해 MSA가 등장하였으며 MSA는 요구들을 만족시키기 위해 다음과 같은 4가지 목적을 가지고 표준화를 진행하고 있다.

□ **불확실성 감소**

공통의 자바 플랫폼을 규격화하기 위한 로드맵을 제시하여 중장기적 불확실성을 감소시키는 것을 목적으로 한다.

□ **복잡도 감소**

Java 2 ME CLDC와 CDC(Connected Device Configuration) 플랫폼 모두에 호환성을 제공하는 아키텍처를 규정하여 복잡도를 감소시키는 것을 목적으로 한다.

□ **호환성 문제 감소**

보안, 어플리케이션 모델 등의 공통의 플랫폼 특성과 의존성을 포함하는 전반적인 플랫폼의 아키텍처를 정의하여 플랫폼의 분열이나 비호환성에 대한 문제를 감소시키는 것을 목적으로 한다.

□ **라이선스 방식 단순화**

MSA에는 매우 다양한 JSR 규격들이 포함된다. 그러나 기존의 방식의 경우라면 이동 통신 사업자나 단말기 제조사는 각각의 JSR 규격에 대한 라이선스를 별도로 진행해야하며 이는 기업들에게 부담을 주게 된다. 따라서 MSA는 MSA에 포함되는 각종 JSR 규격에 관한 라이선스 방식을 단순화하는 것을 목적으로 한다.

3. MSA의 특징

3.1 MSA의 아키텍처

MSA는 JSR-248 규격과 JSR-249 규격으로 구분된다. JSR-248 규격은 CLDC 규격을 하부 플랫폼으로 하고 저/중 사양급 단말에 적용되는 것을 목적으로 한다. 그러나 JSR-248 규격에 CLDC 규격 대신에 CDC 규격을 적용하는 것도 허용하고 있다. JSR-249 규격은 CDC 규격을 하부 플랫폼으로 하고 중/고 사양급 단말에 적용되는 것을 목적으로 하는 JSR-248 규격을 포함하는 상위 규격이다.

MSA의 전체적인 아키텍처는 그림 2와 같다. MSA의 표준화 범위는 컨피규레이션과 프로파일을 포함하는 자바 실행 환경과 공통 API 규격이며 JSR-249 규격은JSR-248 규격에 추가적인 핵심 API 규격과 JSR-232(Mobile Operational Management) 규격을 포함한다.

특히 JSR-249 규격은 JSR-232 규격을 핵심 실행 환경으로 한다. JSR-232 규격은 OSGi Service Platform Mobile Specification Release 4(OSGi MEG) 규격을 채택하고 있고 이동 통신 단말의 원격 관리를 위해 OMA DM(Device Management) 규격을 지원하는 규격을 포함하고 있다.

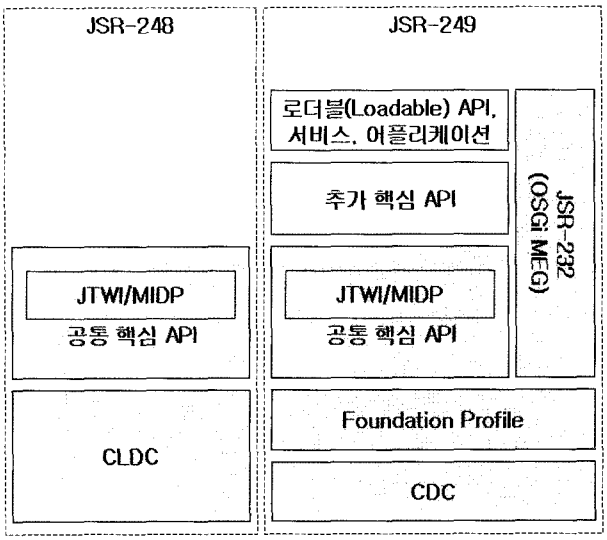

그림 2 JSR-248 및 249 아키텍처

3.2 설계 목적

MSA의 구조는 다음과 같은 3가지의 목적을 가지고 설계되었다.

□ **모바일 자바 환경의 분열 최소화**

MSA의 첫 번째 설계 목적은 예측 가능하고 높은 상호 운용성을 가지는 어플리케이션과 서비스 환경을 정의하여 모바일 자바 환경들의 분열을 최소화하는 것이다. 이 목적을 성취하기 위해 MSA 규격은 필수 및 조건부 필수 JSR 규격, 추가 설명, 추가 요구 사항, 추가 권고 사항을 포함한다.

□ **폭넓은 시장 적용성 보장**

두 번째 설계 목적은 서로 다른 시장과 고객 세분 시장에서 MSA를 적용할 수 있도록 하는 것이다. 이 목적을 수행하기 위해 MSA와 MSA Subset의 두 가지 플랫폼 규격을 정의한다.

□ **규격의 일관성 보장**

세 번째 설계 목적은 JSR-248 규격과 JSR-249 규격 모두에 일관성을 보장하는 것이다.

3.3 JSR-248

JSR-248 규격은 제약적인 자원을 가지고 있으며 지속적인 업그레이드가 필요 없고 어플리케이션에 대

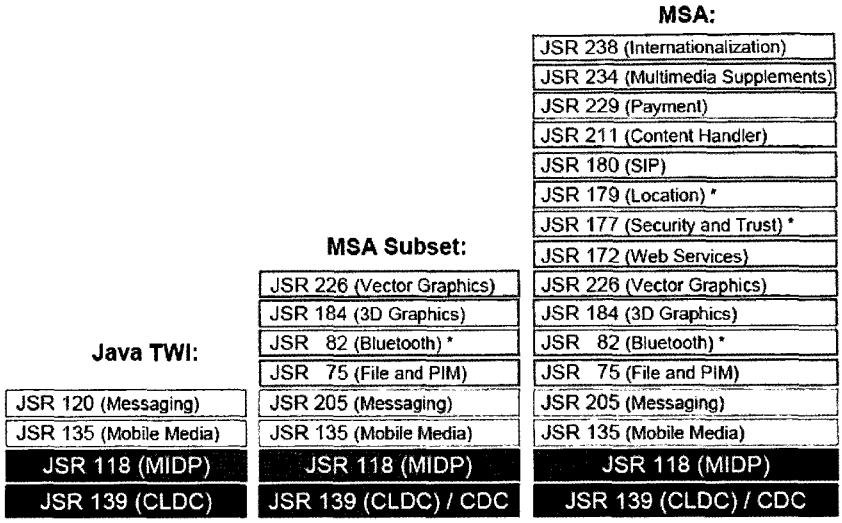

그림 3 MSA 및 MSA Subset의 JSR 규격 집합

한 과금이나 다운로드와 관련된 비즈니스 모델이 단순한 이동 통신 단말을 대상으로 하는 규격이다. 따라서 아키텍처는 핵심 기술과 API에 대한 정적 스택을 미리 정의하고 JTWI 규격에 하위 호환성을 가지는 구조에 초점을 맞추고 있다.

3.3.1 JSR-248 표준화 JSR 규격 집합

MSA 및 MSA Subset 규격은 그림 3과 같은 JSR 규격들로 규정된다.

□ **JSR-139: Connected Limited Device Configuration**

자원이 제약적이고 연결성을 가지는 장치를 대상으로 하는 Java 어플리케이션 플랫폼 규격을 규정한다.

□ **JSR-118: Mobile Information Device Profile**

일반적인 이동 정보 단말(Mobile Information Device)을 위한 Java 실행 환경 규격을 규정한다.

□ **JSR-75: PDA Optional Package**

JSR-75 규격은 PIM(Personal Information Management) 패키지와 FC(File Connection) 패키지의 두 가지 독립된 선택적 패키지들로 구성된다.

PIM API 규격은 PIM 데이터(vCard, vCalender 등)에 접근하는 API 규격을 규정하고 FC API 규격은 이동 통신 단말 또는 메모리 카드의 파일 시스템에 접근하는 API 규격을 규정한다.

□ **JSR-82: Java APIs for Bluetooth**

JSR-82 규격은 Bluetooth 무선 통신 기술을 지원하는 API 규격 집합을 규정한다. JSR-282 규격은 Bluetooth 패키지와 OBEX(Object Exchange) 패키지의 두 가지 선택적 패키지들로 구성된다.

Bluetooth API 규격은 Bluetooth 프로토콜을 이용하여 무선 통신을 지원하기 위한 API 규격을 규정하고 OBEX API 규격은 OBEX 프로토콜을 이용하여 장치 간에 개체를 전송하기 위한 API 규격을 규정한다.

□ **JSR-135: Mobile Media API**

사운드와 멀티미디어를 지원하기 위한 API 규격을 규정한다.

□ **JSR-184: Mobile 3D Graphics API for J2ME**

경량(Light-weight) 3차원 그래픽을 제공하기 위한 API 규격을 규정한다.

□ **JSR-205: Wireless Messaging API**

MMS(Mobile Message Service)등의 메시지 서비스의 송수신을 지원하기 위한 API 규격을 규정한다.

□ **JSR-226: Scalable 2D Vector Graphics API for J2ME**

W3C SVG(Scalable Vector Graphics) 포맷의 렌더링을 지원하기 위한 API 규격을 규정한다.

□ **JSR-172: J2ME Web Services Specification**

JSR-172 규격은 웹서비스 기술을 지원하는 API 규격 집합을 규정한다. JSR-172 규격은 JSR-63 규격에 기반 한 JAXP(Java API for XML Processing) 패키지와 JAX-RPC(Java API for XML-based RPC) 패키지의 두 가지 선택적 패키지들로 구성된다.

JAXP 규격은 XML 데이터를 파싱하기 위한 API 규격을 규정하며 JAX-RPC 규격은 SOAP(Simple Object Access Protocol)에 따른 웹서비스를 지원하기 위한 API 규격을 규정한다.

□ **JSR-177: Security and Trust Services API for J2ME**

JSR-177 규격은 보안 서비스를 제공하기 위한

API 규격 집합을 규정한다. JSR-177 규격은 SATSA-APDU(Application Protocol Data Unit) 패키지, SATSA-JCRMI(Java Card Remote Method Invocation) 패키지, SATSA-PKI(Public Key Infrastructure), SATSA-CRYPTO (Cryptographic) 패키지의 4가지 선택적 패키지들로 구성 된다.

SATSA-APDU 규격은 APDU 프로토콜을 사용하는 ISO-7816-4(Interindustry commands for interchange)에 호환하는 스마트카드를 지원하기 위한 규격을 규정한다. SATSA-JCRMI 규격은 Java Card Client API를 제공하기 위한 규격을 규정한다. SATSA-PKI 규격은 전자 서명 생성과 사용자 증명서를 관리하기 위한 규격을 규정한다. 마지막으로 SATSA-CRYPTO 규격은 Java 2 SE(Standard Edition) Cryptography API 규격의 하위 집합으로 규정 된다.

☐ **JSR-179: Location API for J2ME**

위치 기반 어플리케이션을 위해 위치 정보를 지원하기 위한 API 규격을 규정한다.

☐ **JSR-180: SIP API for J2ME**

SIP(Session Initiation Protocol)을 지원하기 위한 API 규격을 규정한다.

☐ **JSR-211: Content Handler API**

MIME-type 또는 스키마에 기반 하여 URI를 통한 어플리케이션의 실행을 위한 실행 모델과 API 규격을 규정한다.

☐ **JSR-229: Payment API**

과금 트랜잭션 처리를 위한 API 규격을 규정한다.

☐ **JSR-234: Advanced Multimedia Supplements**

JSR-135 규격에 기반 하여 향상 된 멀티미디어 기능을 지원하기 위한 API 규격을 규정한다.

☐ **JSR-238: Mobile Internalization API**

지역화 및 국제화를 지원하기 위한 API 규격을 규정한다.

3.4 JSR-249

JSR-249 규격은 스마트폰이나 PDA 등과 같이 자원에 대한 제약이 적고 높은 처리 능력을 가지며 지속적 업그레이드가 제공될 필요가 있는 폭넓은 비즈니스 모델을 지원하는 이동 통신 단말을 대상으로 하는 규격이다. 따라서 아키텍처는 유연하고 동적인 확장성을 제공하며 JSR-248 규격에 기반 한 어플리케이션을 투명하게 지원할 수 있는 구조에 초점을 맞추고 있다.

3.4.1 JSR-249 표준화 JSR 규격 집합

JSR-249 규격은 JSR-248과 추가 JSR 규격들로 규정을 된다. 단, JSR-249 규격은 CLDC 규격이 아닌 CDC 규격을 하부 플랫폼으로 정의한다. JSR-249 규격은 표 1과 같은 JSR 규격들을 추가로 규정한다.

표 1 JSR-249 추가 규격 집합

JSR	규격명	버전
218	Connected Device Configuration	1.1
219	Foundation Profile	1.1
217	Personal Basis Profile	1.1
232	Mobile Operational Management	1.0

☐ **JSR-218: Connected Device Configuration**

연결성을 가지는 장치를 대상으로 하는 Java 어플리케이션 플랫폼 규격을 규정한다.

☐ **JSR-219: Foundation Profile**

GUI를 요구하지 않으며, 다른 Java 2 ME 프로파일들의 기반이 되는 Java 실행 환경 규격을 규정한다.

☐ **JSR-217: Personal Basis Profile**

기본적인 그래픽 표시를 지원하기 위한 Java 실행 환경 규격을 규정한다.

☐ **JSR-232: Mobile Operational Management**

Java 2 ME CDC 규격에 기반 한 이동 통신 단말의 소프트웨어 컴포넌트 관리 프레임워크를 규정한다.

4. JSR-232

JSR-232 규격은 JSR-249 규격의 핵심 실행 환경을 정의하며 이동 통신 단말에 탑재되는 각종 어플리케이션, 서비스, API와 같은 자바 소프트웨어 컴포넌트의 설치, 업데이트, 실행, 삭제 등의 소프트웨어 생명 주기(Life-cycle) 관리를 가능하게하며 이동 통신 단말에 대한 원격 관리 및 단대단(End-to-end) 상호 운용성을 보장 할 수 있도록 한다.

JSR-232 규격은 시장에서 규격이 빠르게 채택되도록 하고 시장 진입시간을 단축하며 호환성 문제를 최소화하기 위해 OSGi MEG 규격에 기반을 두고 있다. 현재 OSGi MEG 규격은 Public Draft 상태 이고 JSR-232 규격은 Public Review 상태이다. JSR-232 규격은 CDC/FP 규격 기반의 이동 통신 단말의 관리 환경을 규정하며 원격 관리를 위해 OMA DM 규격을 채택하고 있다.

4.1 JSR-232의 특징

4.1.1 프레임워크 계층화 모델

JSR-232 규격의 프레임워크 구조는 계층화 구조이

며 그림 4와 같이 총 4개의 계층으로 구성된다.

□ **실행 환경 계층(Execution Environment)**

실행 환경 계층은 OSGi 서비스 플랫폼을 위한 OSGi 프레임워크와 기본 서비스의 구현을 지원하기 위한 OSGi Minimum 실행 환경과 CDC/FP 규격에 기반을 두는 CDC/FP 실행 환경의 서로 다른 2가지 실행 환경을 규정한다.

□ **모듈 계층(Module Layer)**

모듈 계층은 자바의 모듈화 모델을 정의한다. 모듈 계층은 번들(Bundle) 간의 자바 패키지를 공유하고 은폐하는 규칙을 규정하며 수명주기 계층과 서비스 계층이 없이도 사용 될 수 있다.

□ **생명주기 계층(Life-cycle Layer)**

생명주기 계층은 번들의 수명주기를 관리하기 위한 API 규격을 제공한다. 이 API 규격은 번들의 시작, 정지, 설치, 업데이트, 삭제 방법을 규정한다. 수명주기 계층은 모듈 계층을 요구한다.

□ **서비스 계층(Service Layer)**

서비스 계층은 자바 실행 환경 내에서 동적인 서비스 지향 아키텍처(SOA : Service Oriented Architecture)를 규정한다.

□ **보안 계층(Security Layer)**

보안 계층은 Java 2 보안 모델에 기반을 두고 포괄적인 동적 퍼미션(Permission) 모델을 규정한다. 보안 계층은 선택적 계층이다.

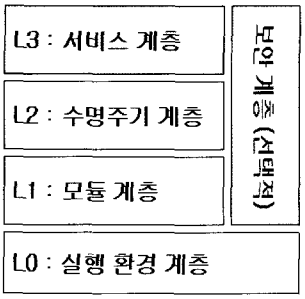

그림 4 프레임워크 계층화 구조

4.1.2 컴포넌트 모델

JSR-232 규격에서 컴포넌트는 번들로써 규정 된다. 번들은 서비스, 어플리케이션, 라이브러리의 컨테이너이며 프레임워크 상에서 실행되는 컴포넌트의 단위로 자바의 Jar (Java Archive) 파일 포맷으로 규정 된다. 번들은 메타 데이터(Manifest)와 다양한 자바 바이트 코드, 네이티브 코드를 비롯하여 이미지, 텍스트 등의 다양한 리소스들을 포함할 수 있고 복수의 서비스를 등록 할 수 있다.

4.1.3 수명주기 모델

번들은 동적으로 설치되고 제거, 시작, 정지 될 수 있다. 그림 5는 번들의 수명주기 전이도이다.

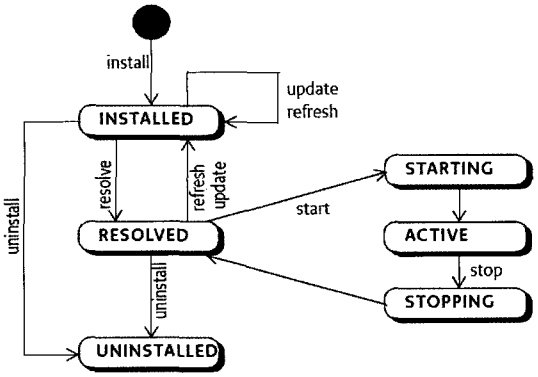

그림 5 번들 수명주기 전이도

번들 상태는 번들의 설치에 성공한 상태를 나타내는 INSTALLED 상태, 설치 된 번들에게 필요한 자바 클래스가 사용가능한 상태를 나타내는 RESOLVED 상태, 번들이 시작되고 있는 상태 STARTING 상태, 번들이 성공적으로 시작 되어 실행되고 있는 상태를 나타내는 ACTIVE 상태, 번들이 정지되고 있는 상태를 나타내는 STOPPING 상태, 번들이 성공적으로 제거되었음을 나타내는 UNINSTALLED 상태의 4가지로 규정 된다.

4.1.4 서비스 지향 아키텍처

OSGi 아키텍처의 핵심 개념은 서비스 지향 아키텍처이다. W3C Web Service Working Group의 Hao He 박사는 서비스 지향 아키텍처를 "상호 작동하는 시스템 사이를 느슨하게 연결(loose coupling)하려는 목적을 가진 아키텍처"라고 정의하고 있다.

서비스 지향 아키텍처는 서비스를 제공하는 서비스 제공자(Service Provider), 서비스 제공자로부터 출판(Publish) 된 서비스를 등록하는 서비스 레지스트리(Service Registry), 등록 된 서비스를 검색하여 사용하는 서비스 소비자(Service Consumer)로 구성 된다. 그림 6은 일반적인 서비스 지향 아키텍처의 개념을 보여준다.

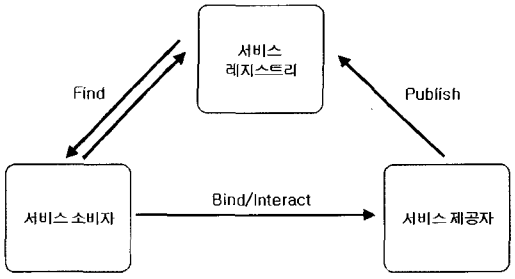

그림 6 일반적인 서비스 지향 아키텍처 개념

일반적으로 서비스 지향 아키텍처를 웹서비스로만 이해하는 경향이 있으나 서비스 지향 아키텍처는 하나의 아키텍처 개념으로 웹서비스에만 국한 된 개념이 아니다. OSGi는 자바 가상 기계(JVM) 내에서 서비스 지향 아키텍처 개념을 실현하고 있다.

OSGi 아키텍처에서 서비스는 자바 언어에서 규정하는 인터페이스로 정의되며 서비스 제공자 번들에 의해 임의의 서비스 속성을 가지고 서비스 레지스트리에 등록된다. 서비스 소비자 번들은 임의의 서비스 속성으로 원하는 서비스를 검색하고 서비스 소비자는 서비스를 통해 서비스 제공자 번들의 구현 메서드를 호출한다. 이와 같은 구조는 컴포넌트 간에 느슨한 연결 구조와 하나의 서비스에 대해 복수의 구현을 가능하게 한다.

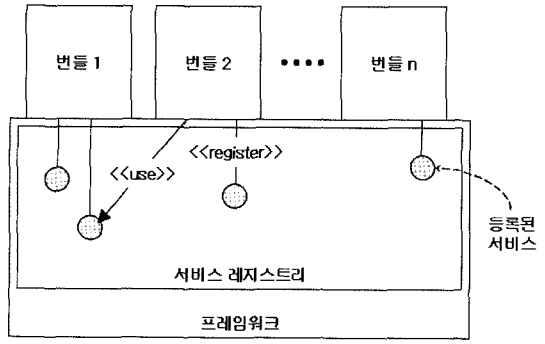

그림 7 OSGi의 서비스 지향 아키텍처

그림 7은 OSGi 아키텍처에서의 서비스 지향 아키텍처의 개념을 보여준다. 그림 7에서 서비스 제공자인 번들 1은 서비스를 프레임워크의 서비스 레지스트리에 서비스를 등록하고 서비스 소비자인 번들 2는 서비스 레지스트리로부터 서비스를 검색하여 번들 1의 구현 메서드를 호출 한다.

4.1.5 관리 모델

기본적으로 OSGi 규격은 원격 관리를 위한 어떠한 프로토콜도 규정하지 않고 있으나 JSR-232 규격은 이동 통신 단말 플랫폼을 원격 관리하기 위한 OMA DM 프로토콜을 규정하고 있다. JSR-232 규격에는 OMA DM 기반의 관리를 위해 DMT (Device Management Tree) Admin 서비스 규격을 규정하고 있다. DMT Admin에 의한 관리 모델은 확장 가능한 트리 모델에 기반하며 각 노드는 플러그인으로 구현 될 수 있다. 관리 트리는 이동 통신 단말 또는 장치에서 가용한 관리 객체를 계층적 트리 구조로 구성하며 각 노드는 유일한 URI로 어드레싱(Addressing)된다.

4.1.6 배포 모델

JSR-232 규격은 이동 통신 단말에 어플리케이션을 배포하기 위한 Deployment Admin 서비스 규격을 규정하고 있다. 이동 통신 단말에 배포되는 어플리케이션 단위를 배포(Deployment) 패키지라고 정의하며 복수의 번들과 각종 자원 및 메타 정보(Manifest)를 포함하는 표준 Jar 파일 포맷으로 규정된다. Deployment Admin은 배포 패키지의 수명주기를 관리하고 설정을 위한 방법을 규정한다.

4.1.7 어플리케이션 모델

JSR-232 규격은 범용(Generic) 어플리케이션 모델과 이종(Foreign) 어플리케이션 모델을 규정한다. 어플리케이션 모델은 OSGi 서비스 플랫폼을 중심으로 서로 다른 이종 플랫폼과 상호작용을 가능하게 하며 통합성과 상호운용성을 제공할 수 있게 한다.

□ 범용 어플리케이션 모델

범용 어플리케이션 모델은 서로 다른 어플리케이션 모델을 하나로 취급할 수 있는 어플리케이션 모델을 제공한다.

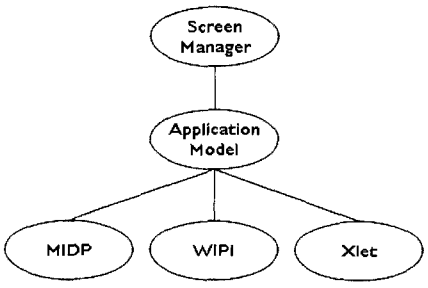

그림 8 범용 어플리케이션 모델

그림 8은 범용 어플리케이션 모델의 개념을 보여준다. 범용 어플리케이션 모델은 사용자에게 사용하는 어플리케이션이 어떤 어플리케이션 모델(OSGi 번들, MIDP Midlet, BREW 어플리케이션, WIPI J-let 등)에 기반을 둔 것인지 알 필요가 없도록 한다.

□ 이종 어플리케이션 모델

이종 어플리케이션 모델은 비 OSGi 어플리케이션이 OSGi의 서비스에 접근하고 서비스를 제공하여 OSGi의 서비스 지향 아키텍처에 편입 할 수 있는 방법을 제공한다. 그림 9는 이종 어플리케이션 모델의 개념을 보여준다.

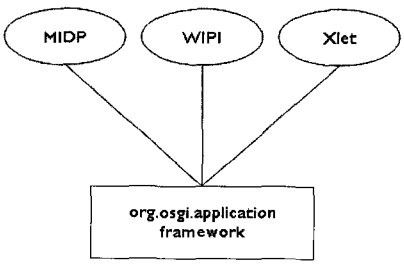

그림 9 이종 어플리케이션 모델

5. 결 론

MSA는 업계의 어플리케이션 실행 플랫폼의 호환성 및 상호운용성 문제를 해결하는 것을 가능하게하며 특히 JSR-232 규격과 JSR-249 규격을 통해 다음과 같은 효과들을 기대할 수 있다.

첫째, 서비스 제공자 또는 이동 통신사들은 기존에 단순히 컨텐츠 수준의 어플리케이션을 제공하던 것에서 벗어나 지속적으로 업그레이드되는 게임 등과 같은 새로운 어플리케이션의 제공과 비즈니스 모델의 확장을 통해 신규 수익 창출 기회를 가질 수 있다.

둘째, 제조사는 원격 관리 모델 및 소프트웨어 원격 배포, 수명 주기 관리를 통해 이동 통신 단말 출시 후에도 원격으로 패치를 제공하거나 진단하여 고객 서비스를 강화 시키고 리콜 문제를 해결 할 수 있다. 또한 개방형 컴포넌트 모델과 서비스 지향 아키텍처 개념으로 소프트웨어 개발에 드는 시간을 단축시키고 비용을 절감 할 수 있으며 자바의 이식성과 모듈화 된 아키텍처는 제품 라인업 확장과 이전에 드는 비용을 감소시킬 수 있다.

셋째, 기업들은 업무용 어플리케이션을 원격으로 직원의 이동 통신 단말에 일괄 배포하고 관리, 변경 할 수 있기 때문에 비즈니스 신속성을 확보할 수 있으며 어플리케이션 개발, 배포, 관리를 비롯하여 이동 통신 단말의 변경 및 선택에 드는 비용을 절감 할 수 있다.

마지막으로 개방성과 표준화, 협업 모델, 동적 다운로드 특성은 이동 통신 단말의 소프트웨어적 기능을 단순화, 표준화 하고 단말의 가격을 낮추는 효과를 가지고 올 것이다. 이는 시장이 서비스와 소프트웨어 중심으로 변화 되고 소비자의 선택성이 높아지게 됨을 의미한다.

MSA는 현재 이동 통신 시장에서 자바의 성공에 기반을 둔 미래의 자바 기반의 이동 통신 서비스를 위한 플랫폼 아키텍처를 제시하고 있으며 2006년 하반기를 기점으로 자바 기반의 이동 통신 시장을 변화 시킬 것으로 예상 된다. 따라서 국내에서도 MSA 표준화 참여와 한국 무선 인터넷 표준인 WIPI 규격과의 협력 및 상호운용성 확보 방안을 고려해야할 것이다.

참고문헌

[1] OSGi Alliance, "OSGi Service Platform Core Specification Release 4", 2005.8.

[2] OSGi Alliance, "OSGi Service Platform Service Compendium Specification Release 4", 2005.8.

[3] OSGi Alliance, "OSGi Service Platform Mobile Specification Release 4, Draft", 2005.11.

[4] Java Community Process, "Java Specification Request 248, Mobile Service Architecture Specification, Proposed Final Draft Version 0.95", 2006.4.

[5] Java Community Process, "Java Specification Request 249, Mobile Service Architecture Advanced", 2004.

[6] Java Community Process, "Java Specification Request 232, Mobile Operational Management for Java 2 Platform, Micro Edition, Version Pre-1.0, Draft 0.9, Public Draft Review", 2006.6.

[7] Java Community Process, "Java Specification Request 185, Java Technology for the Wireless Industry Specification Version 1.0", 2003.6.

[8] Asko Komsi, Mark Düsener, "Mobile Service Architecture Initiative: The Latest News on JSR 248 and 249" 2005 JavaOne Conference, 2005.5.

[9] Bill Day, "J2ME Platform at Five: Where We've Been, and Where We'll Be at Ten", 2004 JavaOne Conference, 2004.6.

[10] Victor Brilon, "Mobile Service Architecture", 2005 Forum Nokia Tech Days", 2005.8.

[11] Kay Glahn, Asko Komsi, "Mobile Service Architecture initiative: MSA Hitting the Market Soon", 2006 JavaOne Conference, 2006.5.

[12] Jon Bostrom, Gabor Pecsy, "Mobile Java Technology JSR232 Architecture and Benefits", 2006 JavaOne Conference, 2006.5.

[13] 한국 무선 인터넷 표준화 포럼, "모바일 표준 플랫폼 WIPI 2.0.1", 2004.9.

[14] 강성천, "OSGi 기술 표준화 동향" 2006 IT 포럼 코리아 컨퍼런스, 2006.4.

강 성 천

1998 광운대학교 전자통신공학과(학사)
2000 광운대학교 전자통신공학과(석사)
2001 에크로넷(주) 인터넷팀 부팀장/
OSGi 프로젝트 리더
2001~현재 커넥티드시스템즈 대표이사
관심분야 : 개방형 서비스 플랫폼, 유틸리티 컴퓨팅, 유비쿼터스 컴퓨팅, 디바이스 네트워킹
E-mail : sckang@connectedsys.com