모바일 환경용 Java 컨피규레이션 API 구현 및 테스팅

전준근°, 김현수
충남대학교 전기정보통신공학부 컴퓨터전공
{jgjeon, hskim}@ce.cnu.ac.kr

Implementation and Testing of Java Configuration APIs for Mobile Environments

Jun-Geun Jeon°, Hyeon Soo Kim
Dept. of Computer Science and Engineering, Chungnam National University

요 약

휴대용 단말기나, PDA, 양방향 호출기 등 소형 모바일 디바이스를 이용한 무선 인터넷 서비스 사용이 늘어가고 있다. 이러한 디바이스들은 기존의 데스크탑 컴퓨터와는 달리 시간과 장소에 구애됨이 없이 언제, 어디서나 디바이스들 간의 통신이 가능하다. Sun사는 이러한 모바일 디바이스를 위해 J2ME 플랫폼을 발표하였다. J2ME는 KVM 및 디바이스에 맞추어진 실행환경 API 집합으로 이루어진 컨피규레이션, 프로파일 등으로 구성된다. 그러나 Sun사는 고가의 license royalty 정책을 펴고 있다. 이러한 이유로 Sun사의 license royalty 정책을 피할 수 있는 독립적인 플랫폼 개발이 절실히 요구된다. 본 논문에서는 J2ME 컨피규레이션 API를 독립적으로 구현하여 모바일 환경에 적합한 플랫폼의 일부를 구현한 내용과 API 테스팅 및 통합 테스팅 환경의 구축과 관련된 내용들을 기술한다.

1. 서 론[1)]

최근 들어 무선 인터넷 서비스와 모바일 디바이스의 확산으로 인해 모바일 시장의 규모가 급격히 커지고 있다. 주위를 둘러보면 휴대폰이나 PDA 등의 모바일 디바이스를 이용하여 무선 인터넷에 접속한 후 다양한 서비스를 사용하는 예를 자주 볼 수 있다.

Sun사는 1999년 모바일 디바이스와 같은 메모리 제약형 임베디드 시스템을 위한 Java 플랫폼인 J2ME(Java 2 Platform, Micro Edition)를 발표하였다. J2ME의 핵심적인 요소는 자바 플랫폼의 핵심 기술에 대한 명세인 컨피규레이션(Configuration)과 특정 디바이스 및 산업군에 맞는 API를 제공하는 프로파일(Profile)이다.

그러나 Sun사는 고가의 license royalty 정책을 펴고 있다. 즉 Sun사가 제공하는 J2ME API들을 사용하여 모바일 환경에서 동작하는 상업용 어플리케이션 프로그램을 개발한다면 Sun사에 비싼 royalty를 지불하여야 한다. 이러한 이유로 Sun사의 license royalty 정책을 피할 수 있는 독립적인 플랫폼 개발이 절실히 요구된다. 본 연구에서는 Sun사의 CLDC(Connected Limited Device Configuration) 명세를 따르지만 실제 구현은 Sun의 소스 코드와 무관하게 독립적으로 구현하는 크린룸(Clean Room) 형태로 컨피규레이션에 해당하는 기능들을 개발한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 J2ME의 구조에 대해 살펴보고, 3장에서는 컨피규레이션 API 클래스 라이브러리의 구현에 대해서 설명한다. 4장에서는 구현한 API 클래스의 테스팅 환경을 구축하고, 5장에서는 결론 및 향후 과제에 대하여 기술한다.

2. J2ME 구조

1999년에 팜 OS용으로 KVM(Kilobyte or Kauai Virtual Machine)이 소개된 이후 소형기기를 위한 자바 플랫폼이 정의되기 시작했고 J2ME가 만들어 졌다. J2ME는 설계 시 소형장비들의 다양한 특징을 각각 활용할 수 있으면서 동시에 자바가 제공하는 특징인 일반성을 최대한 활용할 수 있도록 노력하였다. 대부분의 소형기기는 통일된 GUI를 제공하지 않는데 J2ME에서는 가능한 일반적인 GUI를 제공하려고 노력하고 자바의 기본 특징인 플랫폼 독립성을 유지하려고 하였다. J2ME에서는 향후 무선 환경에서의 동작을 고려하여 네트워킹 기능을 추상화하는데 주력하였다. 이로 인해 코드가 다른 기종에 쉽게 이식 가능하다. 또한 지속성(persistence) 기능을 통해 네트워크가 연결되어 있지 않은 동안은 로컬 캐싱이 되도록 하였다.

J2ME는 어플리케이션이 실행되기 위한 환경으로 Java 가상 머신인 KVM과 디바이스에 맞추어진 실행환경 API 집합으로 이루어진 컨피규레이션, 프로파일 등으로 구성된다.

그림 1은 Java 2 Platform을 보여 주고 있는데 여기서 오른쪽에 표시한 부분에서 J2ME의 구성을 볼 수 있다. 컨피규레이션은 비슷한 규모의 소형기기들이 공통으로 가지고 있어야 할 API의 집합을 정의하고 프로파일은 특정한 그룹의 소형기기의 특징을 반영한 API의 집합을 말

1) 본 연구는 한국과학재단 지정 충남대학교 소프트웨어연구센터의 지원에 의한 것입니다.

한다.

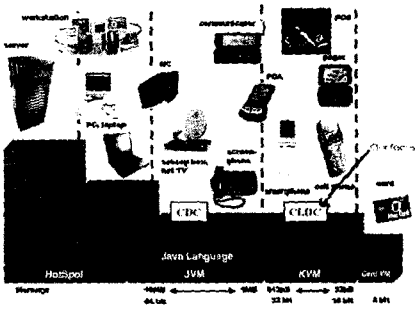

그림 1. Java 2 Platform

3. 설계 및 구현

컨피규레이션 CLDC API에 들어 있는 대부분의 클래스들은 J2SE API에 똑같이 포함된다. 이러한 클래스와 인터페이스들은 실제로 일반 Java 프로그램에서 보았던 것과 같은 것들이다. CLDC API의 이러한 부분은 java.lang이나 java.util과 같이 낯익은 J2SE 이름으로 패키지 내에 위치하게 된다.

J2SE로부터 직접 가져온 클래스와 인터페이스 외에도 몇 개의 인터페이스가 CLDC API에 추가적으로 들어 있다. 이러한 인터페이스는 주로 네트워크를 다루는데, 이것은 네트워크 장치에 맞도록 크기를 줄이기 어려운 J2SE API의 부분이다. 즉 장치가 네트워크 통신을 이용하는 경우, J2SE에서의 네트워크 지원 기능은 너무 광범위하기 때문에 장치에 많은 부담을 주게 된다. 또한 네트워크 장치의 요구사항도 장치에 따라 변화가 심하다. 따라서 일련의 장치에 대한 특정한 네트워크 클래스와 인터페이스를 컨피규레이션이 아닌 프로파일에서 지정하는 것이 훨씬 더 유리하다. 이러한 이유로 CLDC는 일반적인 네트워크를 이용하는 인터페이스만을 정의하고, 이러한 인터페이스를 이용하는데 필요한 자세한 사항들은 프로파일에 맡기게 된다.

CLDC는 모두 4개의 패키지로 구성되며 각각은 java.lang, java.io, java.util, javax.microedition.io이다. 이중에서 java.lang, java.io, java.util은 CLDC 환경에 맞는 메소드 등을 J2SE로부터 물려받아 사용한다. javax.microedition.io 패키지는 CLDC에서 새롭게 정의한 클래스들이다.

기본적으로 컨피규레이션 API 클래스 구현은 CLDC 명세를 준수하여 구현하였다. 설계시 큰 어려움은 없었으나 명세에 나와 있지 않은 숨겨진 클래스를 구현해야 하였고, native 메소드와 관련된 코드의 구현이 고려사항으로 제기 되었다.

숨겨진 클래스는 주로 국제화(internationalization)와 입출력에 관련한 것들인데 CLDC에서 공통적으로 사용되는 기능을 별도의 클래스로 구현하여 따로 패키지화하였다.

native 코드와 관련해서는 KVM 계층에서 구현되어야 되는 것이 대부분이므로 KVM 구현 팀과 상의하기로 하였다.

4. 테스팅 환경의 구축

테스팅 환경은 API 테스팅과 통합 테스팅(integration testing) 환경을 구축하였다.

API 테스팅 방법은 각 패키지의 API 클래스들을 하나하나 독립적으로 호출하여 테스팅하는 방법이다. 통합 테스팅 방법은 J2ME 에뮬레이터를 이용하는 것인데 에뮬레이터에서 사용하는 CLDC 부분을 본 논문에서 구현한 컨피규레이션으로 대체하고 MIDlet 프로그램을 이용하여 테스팅하는 방법이다.

4.1 API 테스팅 환경

API 테스팅은 설계의 최소 단위인 API 클래스에 대해 테스팅을 수행하는 방법이다. TestDriver는 본 논문에서 구현한 컨피규레이션 클래스인 IUT(Implementation Under Testing) 클래스를 호출하고 결과 값을 받아 출력하게 된다. TestDriver는 테스트하고자 하는 IUT 클래스 모듈에 맞게 테스트 케이스를 하나하나 적용하여 각 클래스에 대한 테스트 결과를 얻을 수 있다.

테스트 결과가 바른지 혹은 틀린지를 평가하기 위해서 본 연구에서는 IUT 클래스와 이 클래스에 대응되는 CLDC 참조 클래스를 함께 호출하는 방법을 사용하였다. IUT 클래스에 대응되는 CLDC 참조 클래스는 Sun사에서 이미 구현한 CLDC 클래스를 의미한다. 이 방법은 IUT 클래스와 대응 CLDC 참조 클래스는 모두 J2ME 명세에 따라 구현된 클래스로서 동일 기능을 수행하며, 대응 CLDC 참조 클래스는 Sun사에서 이미 수많은 검증을 거쳤으므로 그 클래스의 수행은 올바른 결과를 나타낼 것이라는 가정을 전제로 하고 있다. 물론 이러한 가정이 항상 참은 아니다. 그렇지만 만일 두 모듈의 호출에서 서로 상이한 결과가 나온다면 두 클래스 중 어느 하나가 오류를 갖고 있음을 가정할 수 있으며, 경우에 따라서는 기존의 Sun사가 개발한 코드에서 오류를 발견할 가능성도 있다. 물론 대부분의 경우 IUT 클래스의 오류일 가능성이 높다. 테스팅의 수행 방법은 다음과 같다. TestDriver는 테스트하고자 하는 IUT 클래스에 적용되는 테스트 케이스를 가지고 IUT 클래스와 대응 CLDC 참조 클래스를 함께 호출하고, 대응 CLDC 참조 클래스의 수행 결과를 테스트 오라클로 사용하여 IUT 클래스의 테스트 결과와 비교한다. 이 때 차이가 발생하면 IUT 클래스의 구현에 오류가 있음을 가정할 수 있다. 이러한 과정이 그림 2에 자세하게 묘사되어 있다.

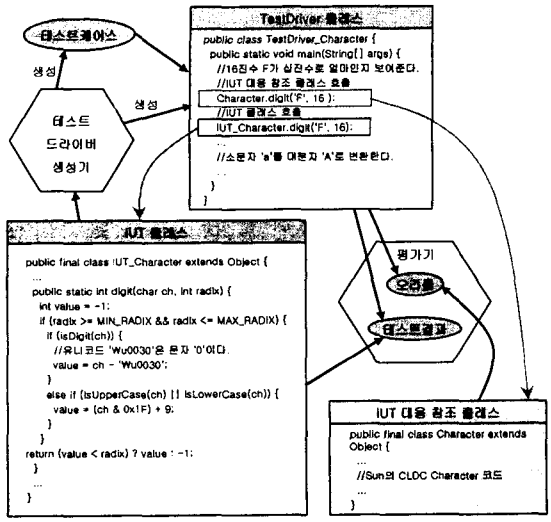

그림 2. API 테스팅

그림 3은 위의 API 테스팅 환경으로 Character 클래스를 테스팅한 결과 화면이다.

그림 3. API 테스팅 결과

4.2 통합 테스팅 환경

통합 테스팅의 목적은 API 테스팅이 완벽하게 수행되었더라도 클래스간 또는 다른 계층들 간에 인터페이스 오류가 있을 수 있으므로 이를 찾기 위해 필요하다.

통합 테스팅환경을 구축하기 위해 표준 J2ME 에뮬레이터인 Wireless Toolkit을 사용하였다. J2ME 에뮬레이터는 데스크탑 PC에서 MIDlet을 실행시켜 실제 장치에서 MIDlet이 어떻게 실행되는가를 보여주는 도구이다.

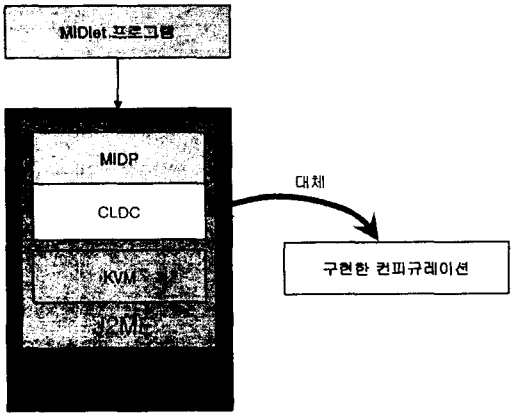

그림 4. 통합 테스팅 환경

그림 4는 통합 테스팅 환경을 보여준다. 그림 4와 같이 J2ME 에뮬레이터에서 사용하고 있는 CLDC를 구현한 컴피규레이션으로 대체해서 테스팅 한다. 모바일 컴퓨팅 환경에서 실행되는 MIDlet 프로그램을 실행함으로서 구현한 컴피규레이션을 통합 테스팅 할 수 있다.

에뮬레이터는 ..₩lib₩midpapi.zip에 CLDC/MIDP 관련 클래스를 패키지화해서 사용한다. midpapi.zip의 기존클래스는 남겨두고 에뮬레이터의 CLDC 클래스들을 본 논문에서 구현한 클래스들로 모두 대체하여 MIDlet 프로그램을 구동시켜 가며 테스팅을 수행한다. 물론 이 때 MIDlet 테스팅 프로그램들은 본 논문에서 구현한 테스팅 환경에 의해 자동 생성된다.

그림 5는 대체한 클래스 상에서 실행한 결과 화면이다. 이런 방법을 통해 Sun사가 개발한 컴피규레이션을 실행했을 때와 똑같은 결과를 얻을 수 있었다.

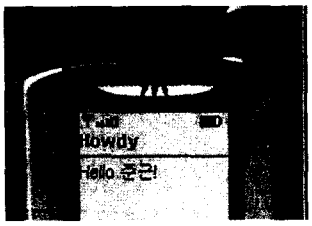

그림 5. 통합 테스팅 결과

5. 결론 및 향후 과제

본 논문에서는 CLDC 명세서 1.0a를 참조하여 모바일 환경용 컴피규레이션 API를 구현하였다. 또한 본 연구에서는 구현한 클래스들을 테스팅하기 위해 API 테스팅과 통합 테스팅 환경을 구축하였다. 하지만 컴피규레이션 하나만 가지고는 J2ME의 모든 기능을 사용할 수 없다. 따라서 향후과제로 KVM과 MIDP를 구현해야하고 이들을 이미 구현한 컴피규레이션과 연동시켜야 한다. 그런데 KVM은 다른 팀에서 현재 개발 중 이므로 본 연구에서는 MIDP의 개발에 치중하여야 한다. 또한, 이들이 모두 개발되고 나면 이들 간에 연동 과정을 테스팅하기 위한 연동 테스팅 방법을 고려해야 할 것이다.

참 고 문 헌

[1] Sun Microsystems Inc., CLDC Specification Version 1.0a, 2000

[2] Kim Topley, J2ME™ in a Nutshell, O'Reilly, 2002

[3] Eric Giguere, Java™ 2 Micro Edition, John Wiley & Sons, Inc, 2000

[4] 권기경, 박용우, IT EXPERT 모바일 자바 프로그래밍, 한빛미디어(주), 2002

[5] Joseph L. Weber, Special Edition Using Java™ 2 Platform, (주)교학사, 2000