J2ME 플랫폼 기반의 테스트케이스 생성 기법

김 상 일[†] · 노 명 기[††] · 류 성 열[†††]

요 약

모바일 소프트웨어의 생산성을 높이고 신뢰성 있는 소프트웨어를 개발하기 위한 모바일 소프트웨어 테스트의 중요성이 부각되고 있다. 모바일 소프트웨어의 테스팅 기술이 효율적으로 적용되기 위해서는 모바일 플랫폼 기반의 테스트를 자동화 할 수 있는 기술이 필요하다. 즉, 모바일 플랫폼에서 제공하는 API를 테스트 하기 위한 테스트케이스를 생성할 수 있는 기법이 필요하다. 테스트케이스를 생성하게 되면 소프트웨어의 생산성과 신뢰성을 향상 시킬 수 있을 뿐만 아니라, 테스트 기간 및 비용을 줄일 수 있다는 이점이 있다. 본 논문에서는 기존의 테스트 자동화에 관련된 연구와 테스트 자동화 도구에 대한 연구를 통해 테스트케이스 생성 범위를 설정하고, J2ME 플랫폼에 사용될 수 있는 테스트케이스 생성 기법인 키워드 방식 기법(Keyword Driven Method)을 제시하였으며, 제안한 생성 기법을 적용하여 J2ME 플랫폼 기반의 테스트에 활용 가능한 테스트케이스 생성 기법이 되도록 하였다.

키워드 : 소프트웨어 테스팅, 임베디드 소프트웨어 테스팅, 모바일 소프트웨어 테스팅, 모바일 응용프로그램 테스팅, J2ME 소프트웨어 테스팅, 테스트케이스 생성

A Test Case Generation Techniques Based on J2ME Platform

Sang Il Kim[†] · Myong Ki Roh[††] · Sung Yul Rhew[†††]

ABSTRACT

The importance of mobile software test is being addressed to improve the productivity and reliability of the software. Test automation technique based on mobile platform is required for effective application of mobile software test. That is, a technique is needed to generate test case for mobile platform API. When test case generated, software productivity and reliability are improved, while test duration and cost are decreased. In this paper, we identified test case generation scope through previous works about test automation, suggested keyword driven method, a test case generatinon technique on J2ME platform, and recognized that proposed method can be applicable to generating test case based on J2ME platform.

Key Words : Software Testing, Embedded Software Testing, Mobile Software Testing, Mobile Application Testing, J2ME Software Testing, Test Case Generation

1. 서 론

모바일 소프트웨어 개발의 비중이 점차 높아져 감에 따라 산업체 전반에 걸쳐 모바일 소프트웨어의 중요성이 매우 중요한 위치를 차지하고 있다. 모바일 소프트웨어는 일반 소프트웨어와는 달리 고신뢰성, 소용량 메모리, 실시간 작동을 요구하는 품질 특성을 갖는다. 따라서 모바일 소프트웨어의 생산성을 높이고 신뢰성 있는 소프트웨어를 개발하기 위해서는 모바일 소프트웨어를 테스트하기 위한 "테스팅 기술"이 필요하다. 또한 모바일 소프트웨어의 테스팅 기술이 효율적으로 적용되기 위해서는 테스트케이스(TestCase)를 생성할 수 있는 기술이 필요하다.

지금까지 연구된 테스트 자동화와 자동화 도구 개발의 대부분은 테스트케이스를 수행하고 수행 결과를 모니터링 하는 과정에 중점을 두고 있기 때문에 테스트케이스는 소프트웨어 개발자 및 테스트 담당자에 의해서 직접 생성되어야 했다. 테스트 기법에 대한 많은 연구가 진행되었지만, 소스코드나 명세로부터 직접 테스트케이스를 생성하는 것은 일반 개발자에게는 여전히 어려운 작업이다. 또한 개발자가 직접 생성하는 테스트케이스들은 보통 정상적인 시나리오 위주로 만들어지기 때문에 다양한 테스트시나리오에 대한 검증이 부족하게 되는 문제점이 있다. 따라서 코드기반의 테스트케이스를 생성 하게 되면 테스트 준비 과정의 편의성을 크게 향상 시킬 수 있을 뿐만 아니라 테스트 기간 및 비

※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.
† 준 회 원 : 숭실대학교대학원 컴퓨터학과
†† 준 회 원 : 숭실대학교대학원 컴퓨터학과
††† 종신회원 : 숭실대학교 정보과학대학 컴퓨터학부 교수
논문접수 : 2005년 7월 4일, 심사완료 : 2005년 12월 6일

용을 단축시킬 수 있는 이점이 있다. 그러므로 모바일 소프트웨어의 개발은 일반 데스크탑 소프트웨어보다 비교적 개발기간이 짧고, 시장 진입 시간이 짧기 때문에 테스트케이스의 생성은 필수적이다.

본 논문에서는 기존의 테스트에 관련된 연구와 테스트 도구에 대한 연구를 통해 테스트케이스 생성 범위를 설정하고, J2ME 플랫폼에 사용될 수 있는 테스트케이스 생성 기법을 연구하였다.

본 논문의 구성은 2장에서 관련 연구로서 다양한 무선 인터넷 플랫폼 중에서 Sun의 J2ME 플랫폼을 소개한다. 테스트케이스를 생성하기 위한 키워드 방식 기법(Keyword Driven Method) 을 소개하고, 이를 응용하기 위해 J2ME 플랫폼을 분석하고, J2ME 플랫폼에서 지원하는 API에 키워드 방식 테스트 기법(Keyword Driven Test Method)을 살펴본다. 3장에서는 테스트케이스를 생성하는 기법을 지원하는 도구를 설계하고 구현한다. 본 논문에서 제시한 키워드 방식 기법을 응용한 테스트케이스 생성 기법은 코드 기반에 의한 테스트케이스 생성이고, 범위는 테스트프로세스 중 테스트 설계와 테스트 구현 단계의 일부로 한정하였다. 4장에서는 제안한 테스트케이스 생성 기법을 사례연구를 통하여 평가하며, 마지막으로 5장에서는 본 논문의 결론을 맺는다.

2. 관련 연구

2.1 테스트케이스 생성 기법[6, 7]

일반적으로 테스트 생성 과정은 기술집약적인 과정이고 테스트 수행 과정은 노동집약적인 과정이다. 현재까지 이 분야의 연구는 노동집약적인 테스트 수행 과정의 자동화로 편중되어 있다[4]. 특히 테스트 생성 과정의 하나인 테스트케이스 설계는 거의 무한히 많은 테스트케이스가 생성될 수 있으므로 테스트 담당자가 테스트에 적합한 테스트케이스를 설계하는 것은 매우 어려운 작업이다. 테스트케이스는 테스트 입력 값, 테스트시나리오, 그리고 예상 값으로 구성된다. 테스트케이스 설계의 시작점인 테스트 입력 값, 즉 테스트 데이터를 생성하면 이 작업을 효율적으로 할 수 있게 된다.

테스트케이스를 생성하는 방법으로는 다음과 같은 것들이 있다.

명세 기반 생성 기법은 소프트웨어 기능 명세로부터 테스트케이스를 생성하는 기법이다.

코드 기반 생성 기법은 구현된 코드로부터 테스트케이스를 생성하는 기법이다.

캡처(Capture) 기법은 시험 하고자 하는 프로그램을 사용자가 시범적으로 사용하고 이때 사용 시나리오를 기록함으로써 테스트케이스를 생성하는 기법이다.

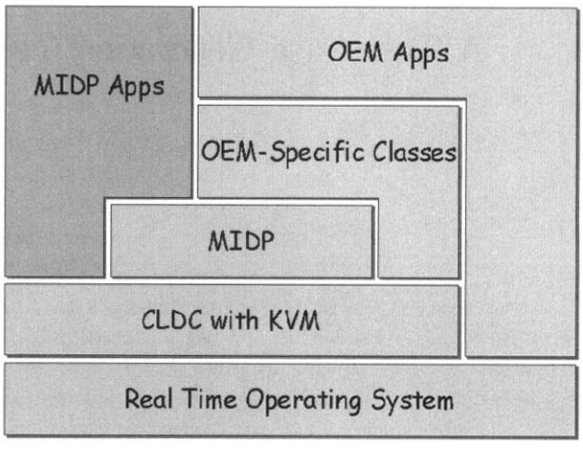
위의 세 가지 방법 중에서 본 논문에서는 테스트케이스를 생성하기 위해서 코드 기반의 생성기법을 적용한다.

2.2 임베디드 소프트웨어 테스팅 도구 아키텍처[8]

본 논문에서는 기존의 자동화 도구에 대한 연구와 모바일 및 임베디드의 특성을 분석하여 모바일 및 임베디드시스템 테스트에 적합한 테스트 지원도구의 아키텍처를 "제조산업을 위한 실시간 임베디드 소프트웨어 테스팅 기술 및 시스템 개발 - 위탁 연구보고서"에서 선택하였으며, 본 논문에서의 연구는 아키텍처 중에서 테스트슈트생성기(TestSuite Generator) 부분에 대한 연구로 한정하였다.

2.3 J2ME(JAVA2 Mobile Edition) 플랫폼의 구조

모바일 응용 프로그램은 CLDC에서 지원하는 가상 머신 위에서 CLDC와 MIDP에서 지원하는 API만을 사용하여 작성되어야 한다. J2ME 플랫폼의 응용 프로그램은 CLDC와 MIDP API에 대한 테스트를 하여 오류가 없으면 실제 J2ME 플랫폼에 탑재되어 있는 모바일 디바이스에서 오류 없이 정상적으로 작동한다. 따라서 본 논문에서의 연구 범위는 MIDP API에 대한 테스트케이스를 생성하는 것으로 한정한다.



(그림 1) J2ME 플랫폼의 구조

〈표 1〉 J2ME 플랫폼에서의 CLDC와 MIDP의 역할

J2ME 플랫폼에서 CLDC의 역할	J2ME 플랫폼에서 MIDP의 역할
- 가상 머신 - J2SE 코어 클래스 입출력 - 네트워크 모델 - 보안 모델 - 국제화 및 지역화	응용 프로그램의 정의 및 제어 사용자 그래픽 환경제공과 이벤트 핸들링 • High-Level 그래픽 API • Low-Level 그래픽 API - RMS를 이용한 데이터 저장 - HTTP를 이용한 네트워킹 - Timer 기능

2.4 키워드 방식(Keyword Driven) 테스트 기법[14-16]

키워드 방식 기법(Keyword Driven Method)은 테스트 계획 방식 기법(Test Plan Driven Method)이라고도 한다. 이 기법은 데이터 방식 기법(Data Driven Method)의 한 가지 방법으로써 기록 후 재생(Record/Playback)기법의 단점을 보완한 기법이다. 이 기법은 특정한 키워드(keyword)를 포함하는 스프레드시트(spreadsheet)를 사용하여 테스터(Tester)에 의해 개발되는 실제의 테스트케이스 문서(TestCase

〈표 2〉 키워드 방식 기법(Keyword Driven Method)의 특징

장점	단점
-모듈화 설계를 이용할 수 있다. -입력값과 데이터 검증에 대한 레코드나 파일을 사용하여 테스트케이스 생성의 중복을 감소시킨다. -테스트스크립트(TestScript)는 응용프로그램 개발 도중에 개발 되어진다. -테스트스크립트는 개별적인 비즈니스 함수에 대해 수행되기 때문에 테스트스크립트는 복잡한 테스트시나리오를 위해 상위레벨에서 쉽게 결합이 가능하다. -상세 테스트 계획(Detail Test Plan) 은 모든 입력값과 검증 데이터를 포함하면서 스프레드시트로 작성된다. 기존의 다른 형태의 테스트케이스 형태를 쉽게 스프레드시트 형태로 변환할 수 있다.	-응용 프로그램이 많은 customized utilities를 사용한다. -테스터는 새로운 키워드와 특정한 포맷을 배워야만 한다. -테스트스크립트 작성 방법을 배워야만 한다. -키워드를 정의하기 위해서는 해당 분야의 전문적 지식이 필요하다.

Documents)를 사용하는 기법이다. 키워드 방식 기법(Keyword Driven Method)에서는 전체 프로세스는 기능성을 포함하고 있는 데이터 중심으로 수행된다. 각각의 키워드들은 프로세싱을 관리한다. 키워드 방식 기법(Keyword Driven Method) 테스트의 장단점은 <표 2>와 같다.

2.5 J2ME 플랫폼의 테스트케이스(TestCase) 생성을 위한 키워드 방식 기법(Keyword Driven Method)의 응용

키워드 방식 기법(Keyword Driven Method)은 테스트 대상 프로그램에서 사용된 함수들의 기능을 키워드(keyword)를 이용하여 정의하고 이를 통해 테스트케이스를 생성한다. 그리고 각각의 생성된 테스트케이스는 데이터를 중심으로 테스트를 한다. 모바일 소프트웨어는 플랫폼에 종속적이기 때문에 모바일 소프트웨어 개발에서 플랫폼이 제공하는 API의 사용은 필수적이다. 그리고 모바일 소프트웨어는 플랫폼에서 제공하는 API의 사용이 반복적일 수밖에 없다. 또한 플랫폼에서 제공하는 API의 클래스와 그 하위의 함수들은 분명한 역할과 기능들을 가지고 있기 때문에 키워드(keyword)로 정의하기가 용이하다. 따라서 J2ME 플랫폼 기반의 소프트웨어를 테스트하기 위한 테스트케이스 생성 기법으로 키워드 방식 기법(Keyword Driven Method)이 적당하다.

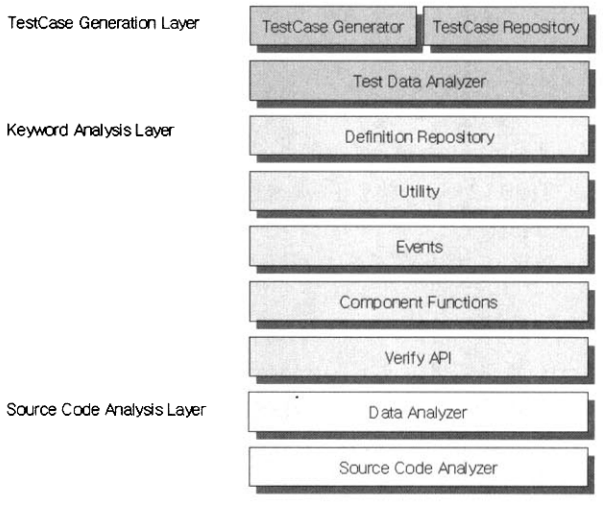
3. 테스트케이스 생성도구의 설계 및 구현

3.1 테스트케이스 생성도구의 아키텍처

본 논문에서는 J2ME 플랫폼의 구조와 플랫폼이 제공하는 CLDC와 MIDP API를 분석하고 키워드 방식 기법(Keyword Driven Method)을 응용하여 J2ME 플랫폼 기반의 응용 프로그램에 적합하고 효율적인 테스트케이스를 생성하기 위한 도구를 설계하였다. 테스트케이스 생성도구의 전체적인 구조는 (그림 2)에서 처럼 크게 세 가지 단계로 구성된다.

소스 코드 분석 계층은 시험대상소스코드(Source Under Test)로부터 J2ME 플랫폼의 API 정보, 이벤트 정보 등의 정보를 추출하고 추출된 정보를 필요한(정의된) 구조로 자료화하여 임시 저장하는 단계이다. 자료화된 정보는 테스트케이스를 생성하기 위한 기초 정보가 된다.

키워드(Keyword) 분석 계층은 테스트케이스 생성도구에서 핵심적인 역할을 하는 단계이다. J2ME 플랫폼에서 제공하는 CLDC와 MIDP API를 분석하여 테스트케이스를 생성하기 위해 미리 정의된 정의 파일을 이용하여 소스 코드 분석 단계에서 추출한 정보와 조합하여 각 단계에 따른 중간 단계의 테스트케이스를 생성하는 역할을 한다.



(그림 2) 테스트케이스 생성도구의 아키텍처

테스트케이스 생성 계층은 테스트케이스 생성 단계는 키워드(keyword) 정의 및 분석 단계에서 생성한 중간 단계의 테스트케이스 간의 연관 관계를 파악하여 이를 다시 분석하여 완성된 테스트케이스를 생성하는 역할을 한다.

3.1.1 소스 코드 분석 계층(Source Code Analysis Layer)

소스코드분석기(Source Code Analyzer)는 파싱(parsing) 기술을 사용하여 시험대상소스코드(Source Under Test)의 구조와 소스코드에 사용된 J2ME 플랫폼의 API 정보, 이벤트 정보 등의 내용을 추출하는 역할을 한다.

데이터분석기(Data Analyzer)는 소스코드분석기(Source Code Analyzer)로부터 추출된 정보를 임시 저장하고, 다음 단계인 키워드(keyword) 정의 및 분석 단계의 클래스들이 사용할 수 있도록 각각의 클래스에 적합하게 정보를 변형하는 역할을 한다.

3.1.2 키워드 분석 계층(Keyword Analysis Layer)

VerifyAPI 클래스는 소스 코드 분석 후 소스 코드에서 사용된 변수와 함수들이 J2ME 플랫폼에서 제공하는 API가 맞는지를 분석하는 클래스이다. VerifyAPI 클래스는 J2ME

플랫폼 API를 파악하기 위해 외부 정의 파일을 사용한다. 외부 정의 파일은 변경과 확장이 용이한 XML 을 이용하여 작성한다. VerifyAPI 클래스는 소스 코드에서 사용된 변수와 함수들을 J2ME API가 맞는가를 분석하여 그 결과를 테스트케이스로 생성하기 위한 기본 입력 값을 만드는 역할을 수행한다.

Component Functions 클래스는 J2ME에서 제공하는 API를 클래스별로 분류하여 각 클래스의 특성과 역할 및 수행할 수 있는 함수를 분석하고 해당 클래스에서 발생될 수 있는 오류를 분석하는 역할을 한다. 예를 들면 J2ME 플랫폼의 High-Level 그래픽 API 중의 TextField 클래스는 사용자로부터 입력을 받아서 그에 따른 결과를 보여주는 기능을 수행한다. 이 때에 발생될 수 있는 오류는 다음과 같다.

- 응용프로그램에서 원하는 타입의 입력 값이 들어왔는가?
- TextField에서 발생될 수 있는 이벤트(event)가 발생되었는가?
- 입력 값에 따른 결과 값이 올바르게 나오는가?

이를 위해서는 각 클래스에 대한 정의가 필요하고, 각 클래스에 대한 정의는 키워드(keyword)로 정의한다. 예를 들면 TextField 클래스에 대한 정의는 <표 3>과 같다. Component Functions 클래스는 소스 코드의 정보와 클래스 정의로부터 초기의 테스트케이스를 만든다.

〈표 3〉 테스트케이스를 생성하기 위한 Component Functions 클래스의 정의

Keyword	Action Performed
VerifyData	현재 TextField에 입력된 값을 검증한다.
VerifyScreen	Event 발생 후에 나타날 화면에 대해 검증한다.
VerifyEvent	TextField에서 발생될 수 있는 Event를 검증한다.

Events 클래스는 J2ME를 포함한 모바일 플랫폼에서는 일반 데스크 탑에서보다 한정된 이벤트(events)를 제공하고 있다. Events 클래스는 타켓 소스 코드에서 사용되는 이벤트(events)를 분석하고 이와 연관되어 있는 컴포넌트들 간의 관계를 분석한다. J2ME에서 제공하는 이벤트(events)는 커맨드 이벤트(Command Event), 키 이벤트(Key Event) 등이 있다. 커맨드 이벤트(Command Event)를 사용할 때 발생할 수 있는 오류는 다음과 같다.

- Command가 속한 컴포넌트에서 Command Event가 발생하였는가?
- Command Event 발생 후에 올바른 절차가 수행되었는가?

이를 위해서 이벤트에 대한 정의를 키워드(keyword)를 이용한다. Command Event에 대한 정의는 다음과 같다. Command 클래스의 인스턴스 cmdOK가 TextFiled tf에 속해있고 입력된 값을 처리하여 그 결과 값을 TextBox tb에 출력하는 명령을 수행하도록 소스 코드에 작성되어 있을 때 <표 4>와 같은 테스트케이스를 생성한다.

〈표 4〉 테스트케이스를 생성하기 위한 Event클래스의 정의

Keyword	Action Performed
VerifyPress	해당 컴포넌트에서 Event가 발생했는가를 검증한다.
VerifyScreen	Event 발생 후 연관된 Screen을 호출하였는가를 검증한다.

Utility 클래스는 J2ME 플랫폼에서 제공하는 API에 대한 직접적인 테스트가 아니라 부가적인 테스트를 하기 위한 클래스이다. Utility 클래스는 타켓 소스 코드에서 사용된 J2ME API에 대한 메모리 사용량, 함수 수행 시간 등을 테스트 하는데 필요한 테스트케이스를 생성하기 위해 데이터를 분석하는 역할을 한다. 예를 들어 특정 함수에 대한 함수 수행 시간을 테스트 하고자 할 때, Utility 클래스는 해당 함수, 루트 함수, 그리고 자식 함수를 구하고 각각에 대한 수행시간을 반환한다.

3.1.3 테스트케이스 생성 계층(TestCase Generation Layer)

테스트데이터분석기(Test Data Analyzer)는 키워드(keyword) 분석 계층의 각각의 클래스에서 생성된 중간 단계의 테스트케이스를 수집하고 분석하여 그들 간의 연관 관계를 파악하는 역할을 한다.

테스트케이스생성기(TestCase Generator)는 테스트데이터 분석기(Test Data Analyzer)에서 수집된 중간 단계의 테스트케이스들과 그 연관 관계를 분석하여 완성된 테스트케이스를 생성하는 역할을 한다. 연관 관계의 예는 <표 5>와 같다.

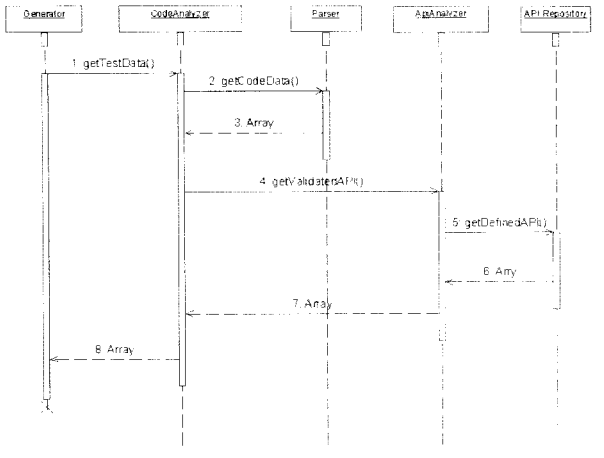
〈표 5〉 Keyword 간의 연관관계

PreAction	Keyword	PostAction
VerifyPress	VerifyScreen	Component
VerifyEvent	VerifyEvent	VerifyPress

3.2 키워드 방식 기법(Keyword Driven Method)을 응용한 테스트케이스 생성 도구의 설계

테스트케이스를 생성하기 위해서는 테스트케이스의 대상이 되는 소스 코드에 대한 정보가 필요하다. 소스 코드 분석 계층은 파싱 기술을 사용하여 시험 대상 소스 코드의 구조와 소스 코드에 사용된 J2ME 플랫폼의 API 정보, 이벤트 정보 등 테스트케이스 생성에 필요한 정보를 분석한다.

소스코드 분석 계층은 소스코드분석기(Source Code Analyzer)와 데이터분석기(Data Analyzer)로 크게 두 부분으로 나누어진다. 소스코드분석기(Source Code Analyzer)는 시험 대상 소스 코드에서 사용된 J2ME 플랫폼의 API 함수를 판별하는 역할을 한다. 데이터분석기(Data Analyzer)는 시험 대상 소스 코드에 사용된 정보를 분석하는 파서(Parser)와



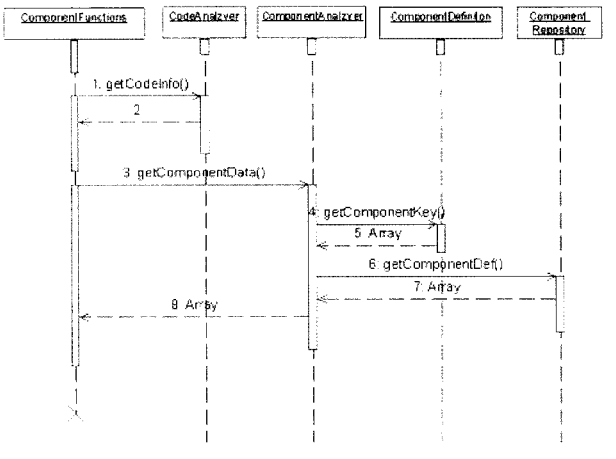
(그림 3) 소스 코드 분석 계층의 시퀀스 다이어그램

API 정보를 분석하는 API분석기(API Analyzer)로 구성된다. 파서는 시험 대상 소스에서 함수를 판별하고, 시험 대상 소스에서 사용되는 자료형의 정의를 파악한다. 그리고 사용된 함수 간의 선후관계 등을 파악하여 테스트케이스를 추출하는데 필요한 정보를 분석한다. API분석기(API Analyzer)는 API 정보를 저장해 놓은 API저장소(Repository)에서 API 정보를 호출하여 소스코드분석기(Source Code Analyzer)가 파서 에서 분석된 정보를 J2ME 플랫폼의 API 함수인지를 판별하는데 도움을 준다. (그림 3)은 이러한 과정을 시퀀스 다이어그램(Sequence Diagram)을 이용하여 나타낸 것이다.

소스 코드 분석 계층이 수행된 후에 키워드 분석 단계가 수행된다.

키워드 분석 계층은 소스 코드 분석 계층에서 판별된 API 함수들을 사전에 정의되어 있는 키워드를 이용하여 매핑시키는 과정이다. 키워드 방식 기법(Keyword Driven Method)은 테스터가 사전에 정의된 키워드(keyword)를 중심으로 스프레드시트 등을 사용하여 테스트케이스 문서를 작성하는 수작업 중심의 기법이다. 이 기법을 자동화하기 위하여 스프레드시트를 대신할 자료구조 형이 필요하다. 본 논문에서는 이를 자동화시키기 위해서 배열형태의 자료구조를 사용하였다. 키워드 분석 계층에서 키워드(keyword)를 이용하여 테스트케이스를 추출하는 과정은 (그림 4)와 같다. ComponentFunctions는 소스 코드 분석 계층에서 분석된 API 함수에 대한 함수에 대한 정보를 컴포넌트분석기(Component Analyzer)에게 요청한다. 컴포넌트분석기(Component Analyzer)는 해당 API 함수에 대한 키워드를 컴포넌트정의(ComponentDefinition)에게 요청하고 해당하는 키워드에 대한 상세한 정보를 컴포넌트저장소(Component Repository)에서 가져온다.

소스 코드에 TextField 타입의 인스턴스 tf가 있고 tf에 대한 테스트케이스는 다음과 같이 생성된다. Component Functions 클래스는 소스코드분석기(Source Code Analyzer) 클래스에서 분석된 정보를 이용하여 소스 코드의 정보와 클래스 정의로부터 <표 6>과 같은 중간 단계의 테스트케이스를 만든다.



(그림 4) 키워드 분석 계층의 시퀀스 다이어그램

〈표 6〉 Component Functions 에서의 중간 단계의 테스트케이스

File	Component	Field	Action(Keyword)
TestSource	TextField	tf	VerifyData
TestSource	TextField	tf	VerifyScreen
TestSource	TextField	tf	VerifyEvent

그런 후에 TextField 타입의 인스턴스 tf와 관련 있는 이벤트 정보를 분석하여 <표 7>과 같은 중간 단계의 이벤트 테스트케이스를 만든다.

〈표 7〉 이벤트(Event)에서의 중간 단계의 테스트케이스

Component	File	Component	Action	Related Components
Command	cmdOK	tf.TextField	VerifyPress	
Command	cmdOK	tf.TextField	VerifyScreen	tb.TextBox

생성된 중간 단계의 테스트케이스들을 바탕으로 각 테스트케이스의 연관관계를 파악하여 최종적으로 <표 8>과 같은 테스트케이스를 생성한다.

〈표 8〉 최종적으로 생성되는 테스트케이스

File	Field	Action	Input Data	Expected Data
TestSample.java	tf.TextField	VerifyScreen	cmdOK.Command	tf.TextField
TestSample.java	tf.TextField	VerifyEvent	cmdOK.Press	TRUE
.....	.....	.....	.....	.....

4. 사례 연구

4.1 개발환경 및 사례연구

본 논문에서 제안하는 'J2ME 플랫폼 기반의 테스트'에 적용하기 위한 키워드 방식 기법은 J2ME 플랫폼의 특징을 제공하기 위해 JAVA 기술을 사용하여 구현하였다. 이 테스트케이스생성기(TestCase Generator)의 구현기술과 개발환경은 다음과 같다.

가. 운영체제 : Windows Server 2003, Windows Server 2000, Windows 2000 Professional, Windows XP Professional

나. 사용언어 : JAVA(JDK 5.0)

이렇게 구현된 테스트케이스생성기를 사례연구를 통하여 적용하였다. 사례연구를 위한 테스트 대상 프로그램은 다음과 같다.

- 이 프로그램은 메뉴 리스트 화면에서 첫 번째 아이템을 선택하고 OK 기능키를 누르면, 데이터 입력창을 보여준다.
- 데이터 입력창에서 데이터를 입력하고 OK 기능키를 누르면 데이터가 저장되고 메뉴 리스트 화면을 보여주고, BACK 기능키를 누르면 데이터 저장 없이 초기의 메뉴 리스트 화면이 나온다.
- 메뉴 리스트 화면에서 두 번째 아이템을 선택하고 OK 기능키를 누르면 저장되어 있는 데이터를 보여준다.

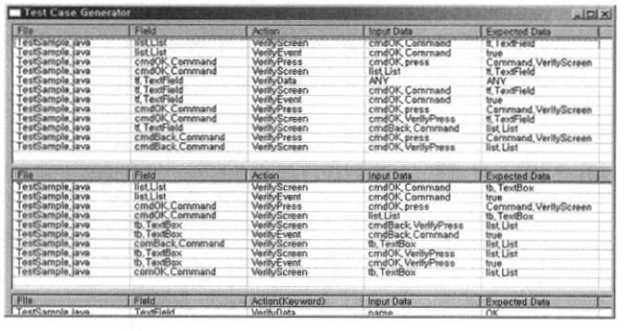
이 프로그램의 소스 코드는 <표 9>와 같다.

본 논문에서 제안한 내용에 따라 구현된 테스트케이스생성기를 이용하여 생성한 테스트케이스의 생성 결과는 (그림 5)와 같다.

〈표 9〉 테스트 대상 소스 코드 예제

```
import java.util.Vector;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class TestSample extends MIDlet implements CommandListener{
        private Display display;
        private List list;
        private Form form
        private TextField tf;
        private TextBox tb;
        private Command cmdOK;
        private Command cmdBack;
        private Vector v=new Vector();
        public TestSample() {
                display = Display.getDisplay(this);
                list = new List("TestSample", List.IMPLICIT);
                        list.append("Data Input",null);
                        list.append("Show Data",null);
                cmdOK = new Command("OK",Command.OK,1);
                cmdBack = new Command("Back",Command.BACK,1);
                list.addCommand(cmdOK);
                list.addCommand(cmdBack);
                list.setCommandListener(this);
                form = new Form("Data Input");
                tf = new TextField("Input", "", 20, TextField.ANY);
                tb = new TextBox("Show Data", "",255, TextField.ANY);
                tb.addCommand(cmdOK);
                tb.addCommand(cmdBack);
                tb.setCommandListener(this);
        }
        protected void startApp() throws MIDletStateChangeException {
                display.setCurrent(list);
        }
        protected void pauseApp(){}
                protected void destroyApp(boolean b) throws MIDletStateChangeException{}
        public void commandAction(Command c, Displayable d)     {
            if(d == list){
                    if(c == cmdOK){
                            if(list.getSelectedIndex()== 0){
                                    display.setCurrent(form);
                            }

            ...<생략>
```



(그림 5) 테스트케이스생성기를 이용해 생성된 테스트케이스

생성된 테스트케이스의 사용 예는 다음과 같다. 테스터는 메뉴 리스트 화면에서 첫 번째 아이템을 선택하고 테스트 입력 값으로 OK 기능키를 누르면 결과 예상 값인 데이터 입력 텍스트 필드가 올바르게 나오는지를 확인하면 된다. 데이터 입력 텍스트 필드가 화면에 올바르게 나타나면 메뉴 리스트의 기능인 OK는 정상적으로 작동한 것이다. 그리고 메뉴 리스트 화면에서 두 번째 아이템을 선택하고 테스트 입력 값으로 OK 기능키를 누르면 테스터는 저장되어 있는 데이터 값을 확인할 수 있는 텍스트 박스가 나타나는지 살펴보면 된다. 텍스트 박스 이외의 컴포넌트가 나타나거나 메뉴 리스트 화면에 그대로 머물러 있으면 OK 기능키는 제대로 작동하지 않은 것이 된다.

4.2 비교실험

본 논문에서 제안한 내용을 바탕으로 구현한 테스트케이스 생성기가 테스트 기간에 소요되는 시간을 감소시킬 수 있다는 사실을 검증하기 위하여 본 논문에서는 몇 가지의 실험을 하였다. 실험 방법은 매뉴얼 방법으로 테스트케이스를 생성하는 시간과 테스트케이스생성기를 이용하여 테스트케이스를 생성하는 데 드는 시간을 측정하여 비교하는 방법을 선택하였다.

테스트케이스생성기에 사용된 언어인 JAVA를 실행시키기 위한 JVM은 하드웨어 적인 성능에 영향을 받을 수 있기 때문에 이에 따른 오차를 최소화하기 위하여 <표 10>에서처럼 다양한 하드웨어 환경에서 실험을 실행하였다.

본 논문에서 수행한 실험의 결과는 다음과 같다. 매뉴얼 테스트 방식으로 테스트케이스를 생성하는 경우(방법 1)와 본 논문에서 제안한 'J2ME 플랫폼 기반의 테스트 기법'을 적용하여 개발한 테스트케이스생성기를 이용하여 테스트케이스를 생성하는 경우(방법 2)에 테스트케이스를 생성하는 시간을 측정하고 비교하였다. 매뉴얼 방식(방법 1)과 테스트

〈표 10〉 테스트 환경

	CPU	메모리	하드 디스크	운영체제
PC 1	3.0 GHz	512MB	80 GB	Windows Server 2003 Enterprise Edition
PC 2	2.0 GHz	256MB	40 GB	Windows 2000 Server
PC 3	1.8 GHz	256MB	40 GB	Windows XP Professional
PC 4	1.7 GHz	128MB	20 GB	Windows 2000 Professional

케이스생성기를 이용한 테스트케이스 생성(방법 2)은 J2ME 플랫폼에서 제공하는 API 중 lcdui 패키지에 있는 각 클래스를 기준으로 하여 기본적인 테스트케이스를 생성하고 테스트케이스를 생성하는 시간을 측정하였다. 테스트케이스생성기를 이용하는 경우에는 하드웨어에 따른 오차 값을 최소화하기 위해서 추가적으로 각 테스트 pc에서 테스트케이스를 생성하는 시간을 측정하고 이에 대한 평균값을 산출하였다. 실험 대상 테스트 소스에서 테스트케이스를 생성하는 시간은 <표 11>과 같다.

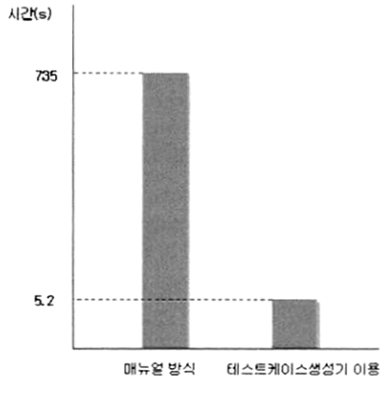
〈표 11〉 테스트 결과 비교

(단위 : second)

방법 / 항목	방법 1	방법 2					비고 (%)
		PC 1	PC 2	PC 3	PC 4	평균	
List Component	60	0.48	0.86	1.02	1.24	0.90	670%
TextField Component	72	0.47	0.84	1.02	1.20	0.88	810%
TextBox Component	70	0.50	0.89	1.05	1.29	0.93	750%
Command Component	70	0.51	0.92	1.08	1.32	0.91	770%
평균값	69	0.49	0.88	1.04	1.26	0.92	750%

실험의 결과에 따르면 본 논문에서 제안한 'J2ME 플랫폼 기반의 테스트 기법'을 적용하여 개발한 테스트케이스생성기를 이용한 방식(방법 2)이 매뉴얼 방식(방법 1)으로 테스트케이스를 생성하는 방식보다 평균 750%의 시간을 절약할 수 있다는 것을 알 수 있다.

위의 실험을 바탕으로 한 다른 실험은 300라인 이하의 소스를 대상으로 두 가지 방식을 이용하여 J2ME 플랫폼에서 제공하는 lcdui 패키지 API에 대한 테스트케이스를 생성하는데 걸린 시간을 측정하였다. 매뉴얼 방식은 한 명이 테스트케이스를 생성하는 작업의 작업시간을 기준으로 하였다. 실험 결과는 (그림 6)과 같다. 실험 결과에 따르면 테스트케이스 생성 시 테스트케이스생성기를 이용하면 크게 시간을 절약할 수 있다는 것을 알 수 있다.



(그림 6) 테스트 비교 평가

5. 결 론

모바일 소프트웨어의 테스트는 사용자 입장에서의 입력이 얼마나 올바르게 수행되는가 하는 것이 핵심이다. 즉, 테스트케이스가 얼마나 정확하게 정의되어 있는가 하는 것이 핵심이다. 사용자의 입력으로는 모바일 폰 에서의 키 누름, 커맨드, 이벤트 등과 같은 것이 있으며, 사용자의 입력에 따른 기능이 완벽히 수행되는 것에 대한 테스트케이스를 생성 해야 한다. 이러한 기능은 J2ME 플랫폼의 API를 이용하여 구현되므로 테스트케이스를 생성하려면 J2ME 플랫폼에서 제공하는 API를 이용하여야만 한다.

본 논문에서는 API 정보를 이용하여 테스트케이스를 생성하기 위해 키워드 방식 기법(Keyword Driven Method)을 제시하였다. 키워드 방식 기법은 테스트 대상 프로그램에서 사용된 함수들의 기능들을 키워드를 이용하여 정의하고 이를 통해 테스트케이스를 생성하는 기법이다. 그리고 각각의 생성된 테스트케이스는 데이터를 중심으로 테스트를 하였다. 모바일 플랫폼에서 제공하는 API의 사용은 반복적일 수 밖에 없으며, 모바일 소프트웨어는 플랫폼에 종속적이기 때문에 모바일 소프트웨어 개발에서 플랫폼이 제공하는 API의 사용은 필수적이다. 또한 모바일 플랫폼에서 제공하는 API의 클래스와 그 하위의 함수들은 분명한 역할과 기능들을 가지고 있어, 키워드로 정의하기가 용이하다.

따라서 본 논문에서는 J2ME 플랫폼에서 제공하는 API들의 역할과 특징을 분석하여 키워드로 정의하고 정의된 키워드를 기반으로 하여 테스트케이스를 생성하는 기법에 대한 내용을 제시하였다.

본 논문에서 제시한 내용을 바탕으로 테스트케이스생성기를 구현하고, 이를 이용한 사례연구를 통해 5인 기준, 10시간 정도의 작업을 테스트케이스생성기를 이용하면 1시간 미만의 시간을 사용하여 작업할 수 있어, 약 9시간 정도의 시간을 절약할 수 있다는 결론을 얻었다.

참 고 문 헌

[1] B. Beizer, 'Software Testing Techniques', 2nd Ed., 1990.

[2] Jone Edvardsson, "A Survey on Automatic Test Data Generation", Proceedings of the Second Conference on Computer Science and Engineering in Linköping, CCSSE'99, ProNova, Norrköping, Oct., 1999.

[3] Marick, Brian, "New Models for Test Development," Proceedings of Quality Week 1999, 1999.

[4] 분산 및 내장형 소프트웨어 테스팅 기술, 정보통신부, 2002.

[5] Jan Tretmans and Ed Brinksma, "TORX: Automated Model Based Testing", European Conference on Model Driven Software Engineering, pp.31-43, Dec., 2003.

[6] Mike Barnett, Wolfgang Grieskamp, Lev Nachmanson, Wolfram Schulte, Mikolai Tillmann and Margus Veanes, "Model-Based Testing with AsmL.NET", European Conference on Model Driven Software Engineering, pp.12-19, Dec., 2003.

[7] N. Goga, "Comparing TorX, Autolink, TGV and UIO Test Algorithms", LNCS 2078, pp.379-402, 2001.

[8] 산업자원부, '제조 산업을 위한 실시간 임베디드 S/W 테스팅 기술 및 시스템 개발-위탁 연구 보고서', 2004.

[9] 홍준성, "모바일 플랫폼의 기술현황 및 발전방향", 정보과학회지, 제22권 제21호, 통권 제176호, pp.8-14, 2004.

[10] '모바일 플랫폼 표준 WIPI', 한국정보통신기술협회, 2002.

[11] 이상윤, 김선자, 김홍남, "한국 무선 인터넷 표준 플랫폼(WIPI)의 표준화 현황 및 발전 방향", 정보과학회지, 제22권, 제21호, 통권 제176호, pp.16-23, 2004.

[12] BREW White Paper, BREW and J2ME-A Complete Wireless Solution for Operators Committed to Java, QUALCOMM.

[13] J2ME Documentation, Available at URL: http://java.sun.com/j2me/docs/index.html

[14] Totally Data-Driven Automated Testing, Available at URL: http://www.sqa-test.com/w_paper1.html

[15] TEST AUTOMATION FRAMEWORKS, Available at URL: http://www.csst-technologies.com/

[16] Data Driven Testing, Available at URL: http://www.wilsonmar.com/datadrvt.htm

[17] Mark Fewster & Dorothy Graham, 'Software Test automation', Addision-Wesley, 1999.

김 상 일

e-mail : hava67@selab.ssu.ac.kr

1998년 숭실대학교 소프트웨어공학과(학사)

2001년 숭실대학교 정보과학대학원 (공학석사)

2003년 숭실대학교 일반대학원 컴퓨터학과 박사수료

관심분야 : 소프트웨어 테스팅, 모바일/임베디드 소프트웨어 테스팅, 소프트웨어개발방법론, 소프트웨어 재사용, 디자인패턴, 리팩토링

노 명 기

e-mail: infinite@selab.ssu.ac.kr

2001년 명지대학교 경영학과(학사)

2005년 숭실대학교 일반대학원 컴퓨터학과(공학석사)

관심분야 : 모바일, 임베디드 시스템, 컴포넌트, 시스템 테스트

류 성 열

e-mail : syrhew@comp.ssu.ac.kr

1981년~현재 숭실대학교 정보과학대학 컴퓨터학부 교수

관심분야 : 리엔지니어링, 소프트웨어 유지보수, 소프트웨어 재사용, 소프트웨어 재공학/역공학, 소프트웨어 테스팅