DOI: 10.3745/KIPSTD.2009.16-D.3.361

J2ME 기반 모바일 응용 소프트웨어 GUI 자동화 테스트 지원기

황 선 명†

요 약

많은 모바일 어플리케이션 개발자들은 매우 엄격한 제약사항 즉 짧은 개발기간, 사용자의 다양한 요구사항 및 잦은 요구 변경에 직면하고 있다. 이러한 개발환경은 사용자가 사용하기 편리한 GUI의 개발과 S/W 개발초기부터 철저한 테스트를 하여 품질을 보증하는 테스트 방법이 필수적이다. 따라서 본 논문은 모바일 응용 S/W GUI를 기존의 매뉴얼을 보고 테스트 하는 단계적이고 수동적인 방법을 탈피하여 사용자 중심의 시나리오를 통하여 자동으로 GUI를 테스트 하는 것을 목적으로 한다. 제시한 테스트 방법은 사용자 중심의 UI를 설계하여 간편하고, 정확하게 테스트 할 뿐만 아니라 테스트케이스를 재사용하는 환경을 지원한다. 제안한 자동화 도구는 시나리오 기반의 정확한 테스트와 재사용성을 높인 테스트 도구로서 기존 테스트 도구와 기능적인 비교를 통하여 성능을 확인하였다.

키워드 : GUI 테스팅, 모바일, 자동화 테스트, J2ME

Test Supporter for GUI of Mobile Application Software in J2ME Platform

Sun-Myumg Hwang†

ABSTRACT

Recently, the most mobile application software developers are being faced by short developing time (fast time-to-market), various requirements and requirement changes. In order to overcome the environment, developer should support user friendly GUI and assure the quality from the early developing process using GUI test method. This paper proposes a test method and tool for mobile software GUI, which reduces test time and supports comfortable test environment by user oriented UI design. We implemented a tool based on test scenarios based image flow for high testing accuracy and test case reusability. And the results show the characteristics of the method compared to the existing tools.

Keywords : GUI Testing, Mobile, Reusability, J2ME

1. 서 론

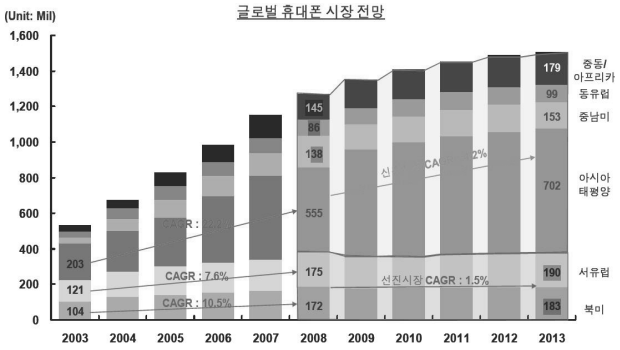
최근 국산 휴대폰은 치열한 경쟁 상황에서 시장적시성을 만족하기 위하여 3~4개월의 짧은 기간에 개발하여 출시함에 따라, 모바일 컨버전스 추세와 함께 응용 서비스 요구가 다양해져서 모바일 S/W의 크기도 매년 20~30% 정도씩 증가함에 따라 S/W 개발이 점점 더 어려운 환경에 처하고 있다[1, 7, 12].

특히, 모바일 S/W는 운영환경이 매우 다양하고, 많은 제약 조건들을 가지고 있어, 모바일 S/W의 테스트보다 어렵다. 모바일 S/W는 호스트에서 개발되고, 다양한 특성들을 가지고 있는 모바일 디바이스에 탑재된다. 따라서 호스트상의 에뮬레이터에서 테스트되고, 최종적으로 타깃 디바이스상에서 테스트되어야 한다.

현재 국내 모바일 디바이스 시장은 지속적인 성장세를 기록하여 2007년 기준 약 4조원의 시장규모에 육박하고 있다. (그림 1)은 글로벌 휴대폰 시장 전망이다. 해가 거듭될수록 휴대폰 시장은 나날이 커져가고 있다[1].

현재 모바일 어플리케이션 개발자들은 클라이언트-서버 및 웹 어플리케이션 개발자와 달리 매우 엄격한 제약사항(메모리, 화면크기, 입력장치 등), 짧은 생명주기, 심한 사용자 편의성에 대한 요구사항 등에 직면해 있다[12]. 신뢰성 있는 모바일 어플리케이션을 개발하기 위해서 개발 초기에

※ 이 논문은 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R01-2008-000-20607-0).
† 종신회원 : 대전대학교 컴퓨터공학과 교수
논문접수 : 2008년 12월 22일
수 정 일 : 1차 2009년 3월 5일, 2차 2009년 3월 11일
심사완료 : 2009년 3월 11일



(그림 1) 글로벌 휴대폰 시장 전망

S/W 결함을 최대한 많이 발견하여 S/W품질을 대폭 향상할 수 있는 방법이 필요하다. 또한 모바일 어플리케이션의 다양한 GUI에 대한 테스트 방법과 구체적인 테스트 케이스 생성 기법이 요구된다[3-6].

모바일 S/W는 데스크탑 S/W보다 단순하고, 크기가 작다. 모바일 환경은 단순한 JVM과 H/W 계층인 디스플레이 장치, 네트워크 포트 등에 접근하기 위한 API를 제공한다. 이러한 환경에서 프로그래밍과 테스팅은 서버 환경보다 더욱더 어렵다. 휴대폰은 각자 다른 H/W 환경, 화면크기와 칼라수, 메모리 크기와 전력 등을 가지고 있다. 또한 같은 어플리케이션이라도 설치되는 타깃 디바이스에 따라 조금씩 다르게 동작 할 수도 있다. 모바일과 데스크탑 S/W 테스트의 차이점은 <표 1>과 같이 요약된다[2, 11].

따라서 본 논문의 구성은 GUI 테스트 지원기의 내부 구조와 각각의 컴포넌트들에 대한 설계 및 설명으로 이루어져 있으며, GUI 테스트 지원기를 실제 구현하여 테스트 하는 방법과 결과에 대하여 설명하고, GUI 테스트 지원기를 적용하였을 때의 적용평가에 대하여 서술하였다.

2. 관련 연구

2.1 모바일 어플리케이션 S/W의 GUI Test

모바일 어플리케이션의 GUI 테스트방법으로 모바일 어플리케이션 S/W 업체에서 가장 많이 사용되는 기술로는 Record/playback, Capture/playback, 명세기반, 베타 테스팅을 사용한다[3-6].

〈표 1〉 모바일과 데스크탑 S/W 테스트 비교

요소	모바일	데스크탑
테스트 기록	GUI 이벤트 기록을 위한 기능을 제공하지 못해, 사용자 인터랙션에 대한 에뮬레이션 의 불가능	GUI 이벤트 기록을 위한 java.awt.Robot 제공
배치 자동화	디바이스 상에 배치와 테스트의 지루한 수작업	배치가 완전히 자 동화되어 있고, 테스트도 비교적 자 동화되어 용이함
테스트 환경 차이점	디바이스 상에 차이점이 있고, 모든 디바이스 상에서 테스트가 요구됨	개발과 배치/테스트가 동일하고, 특정 OS에 의존하는 경우가 거의 없음
API	API 표준화가 아직 미성숙	API 표준이 성숙되어 있고, 다양한 벤더로 부터 JVM 이식 가능

2.1.1 Record/Playback 기법

사용자의 마우스 및 키보드 입력을 통해 선택된 GUI에 발생되는 이벤트를 기록하여 기록된 이벤트를 재생하여 S/W의 GUI 테스트를 수행하는 방법으로 현재 GUI 테스트 도구들은 Record/Playback 기술을 적용하여 많이 개발되어지고 있다. Record/Playback 기법은 단순한 패턴을 가지며, 이 기술을 이용한 테스트 도구를 구현하기가 용이하다. Record/Playback기법의 테스트 도구들은 가능한 짧은 시간 내에 기존 GUI에 대한 테스트 슈트를 생성하고자 할 때 유용하며, 어플리케이션에 대한 변경이 발생하면, 영향을 받는 모든 테스트케이스를 재작성해야 한다. 수작업이 많고 오류에 취약하다는 단점을 가지고 있다[14].

2.1.2 Capture/Playback 기법

시험 하고자하는 모바일 어플리케이션 S/W를 사용자가 시범적으로 사용하고 이때 사용 시나리오를 기록함으로서 GUI의 이상 현상을 기록하는 수행 방법이다[6, 8, 9].

2.1.3 명세기반 테스트 기법

명세 기반 테스트는 시스템의 GUI 명세를 기반으로 시스템을 테스트 한다. 명세 기반 기술 테스트는 다른 테스트 기술에 비교하여 볼 때 여러 조건 요소 들이 있다.

요구사항이나 디자인 명세가 일괄적이어야 하고 완벽해야 하며, 분명하게 기술되는 것을 요구한다. 명세기반 GUI 테스트는 높은 수준의 GUI 명세를 요구한다. 이런 명세를 얻는데 많은 노력이 필요하다.

2.1.4 베타 테스트 기법

베타 테스트는 많은 업체들이 사용하는 GUI 테스트 방법으로서 가장 인기 있는 테스트 방법이다. 베타 버전의 S/W를 출시하여 일반 사용자들에게 GUI 테스트를 수행하게 하는 방법으로 많이 사용되어지고 있으며, GUI 테스트 방법 중 가장 많은 오류를 찾을 수 있는 방법이나 비전문적인 테스터들로 구성되어 그 전문성이 떨어지고 많은 일반 사용자를 이해시키고 교육시키는데 어려움이 따른다.

2.2 J2ME 플랫폼

J2ME 비록 휴대폰이나 임베디드 기기를 위한 플랫폼이지만, 기본적으로 자바가 가지고 있는 특징인, 객체지향방식의 프로그래밍, 코드의 높은 이식성, 안전한 네트워크 보안 지원 및 J2SE와 J2EE와의 상위 호환성은 그대로 유지하고 있다. 여기에 더하여 J2ME는 다음과 같은 특징을 지니고 있다[5, 6, 14].

가. 다중플랫폼호환성(cross-platform compatibility)

100% 순수한 J2ME API를 사용하는 애플리케이션은 쉽

게 다른 업체가 만든 모델과 광범위하게 호환될 수 있다. 단말기가 다르다고 해서 프로그램 개발자 측에서 같은 프로그램을 다시 개발하는 일은 상당히 번거로운 일이다. 하지만 J2ME 플랫폼은 어느 단말기에서든지 시간과 장소에 관계없는 강력한 호환성을 가진다.

나. 보안성

무선 인터넷 환경에서의 보안은 아직도 해결되지 않는 문제점들이 많다. J2ME는 기존 자바의 보안 모델을 무선통신에 적용시킴으로써, 이를 상당부분 해결하고 있다. J2ME로 작성된 애플리케이션은 기본적으로 디바이스의 하드웨어나 다른 리소스에 직접 접근할 수 없기 때문에 바이러스나 다른 악성 프로그램을 만들어낼 수 없다.

다. 동적 애플리케이션 다운로드

무선 서비스를 통한 자바 애플리케이션들은 실시간적으로 동적으로 다운로드 되므로 사용자들은 A/S 센터를 방문하고나, 업그레이드를 쉽게 받을 수 있다.

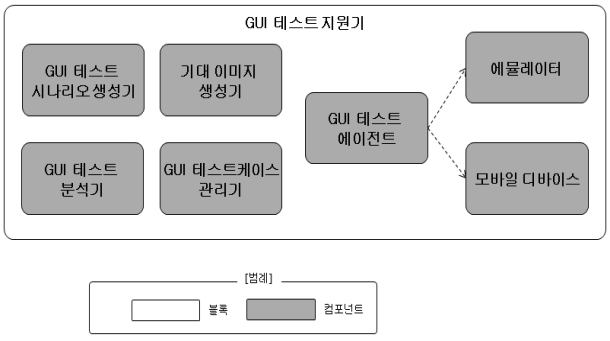
3. GUI 테스트 지원기 설계

GUI 테스트 지원기는 모바일 S/W의 GUI를 테스트 하기 위한 것으로 시나리오에 따라 테스트 케이스를 정의한 후 에뮬레이터 또는 모바일 디바이스에서 동작하는 모바일 S/W를 동작시켜 테스트케이스의 내용과 비교하여 GUI를 테스트 한다[12].

3.1 GUI 테스트 지원기 구조와 기능

(그림 2)는 GUI 테스트 지원기 블록의 구조이다. 블록을 구성하는 컴포넌트의 기능은 다음과 같다.

- GUI 테스트 시나리오 생성기 : GUI를 테스트 위한 시나리오를 작성한다. 테스트 시나리오에는 테스트 개요, 테스트 목적, 주의 사항과 함께 GUI 테스트케이스로 구성된다.
- 기대이미지 생성기 : GUI 테스트케이스에 포함되는 기대이미지를 만든다. 기대이미지는 응용 S/W에서 이벤트를 발생시킬시 출력되는 화면을 미리 정의해논 것이다.



(그림 2) GUI 테스트 지원기 구성도

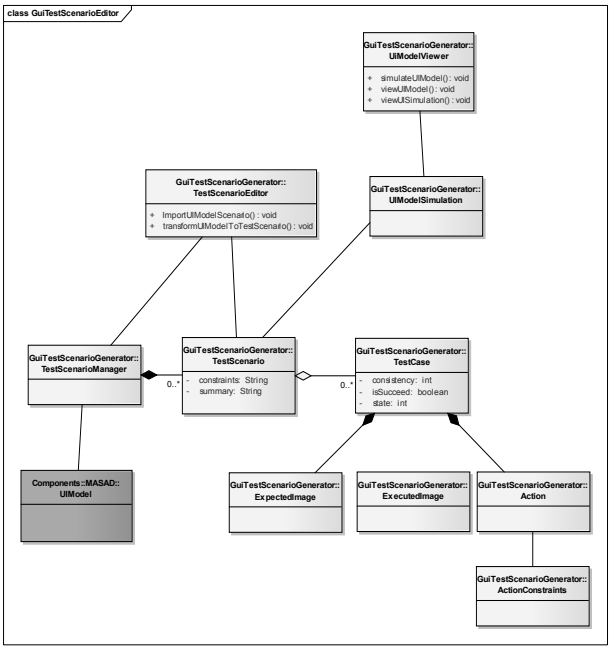
- GUI 테스트 분석기 : GUI 테스트케이스의 기대이미지와 실행 결과 화면을 분석한다.
- GUI 테스트케이스 관리기 : GUI테스트케이스를 관리한다. 에뮬레이터 또는 모바일 디바이스에서 동작하는 모바일 S/W와 연결하여 S/W를 실행시키고 실행 결과를 입력받는다.
- GUI 테스트 에이전트 : 모바일 S/W를 GUI 테스트케이스에 따라 실행하도록 한다. 모바일 S/W와 함께 빌드하여 에뮬레이터 또는 모바일 디바이스 상에서 실행된다. GUI 테스트케이스 관리기에서 보내는 이벤트를 받아서 모바일 S/W를 동작시킨다.

3.2 GUI 테스트 시나리오 생성기

GUI 테스트 시나리오 생성기는 GUI 테스트를 위한 테스트 시나리오를 생성하도록 하는 컴포넌트이다.

테스터가 GUI 테스트를 위해 실행경로를 수행하기 위한 이벤트 순서, 예상화면 등을 만드는 작업은 거의 불가능한 작업니다. 이에 대한 해결책으로 설계도구에서 만든 UI 모델을 이용한다. (그림 3)은 시나리오 생성기의 클래스 다이어그램이다.

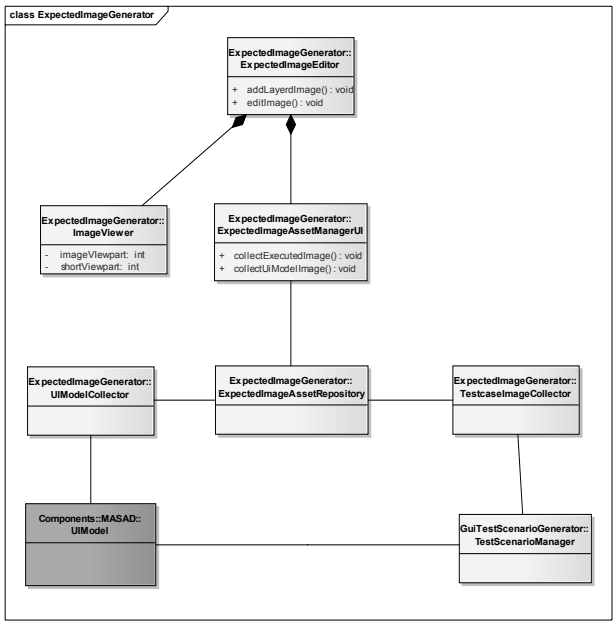
시나리오에는 테스트 개요, 목적, 주의 사항과 함께 GUI 테스트케이스로 구성된다.



(그림 3) GUI 테스트 시나리오 생성기 클래스 다이어그램

3.3 기대이미지 생성기

기대이미지 생성기는 GUI 테스트케이스에 포함되는 이미지를 생성한다. 기대이미지는 모바일 S/W 실행 이후 캡처된 결과 화면과 비교하여 GUI 테스트 성공 여부를 결정하는데 사용된다. (그림 4)는 기대이미지 생성기의 클래스 다이어그램이다.



(그림 4) 기대이미지 생성기의 클래스 다이어그램

■ ExpectedImageAssetManagerUI : 후보 기대이미지를 관리하기 위한 관리기 화면이다. 기대이미지 자산리파지토리 내용을 사용자에게 보여주고 선택할 수 있도록 한다.

■ ExpectedImageAssetRepository : 기대이미지 자산을 관리한다. 기대이미지 자산은 UI모델로부터 정의한 화면과 실행화면, 기존에 만든 기대이미지들로 구성된다.

■ ExpectedImageEditor : 기대이미지를 생성하기 위한 메인화면이다. 기대이미지를 편집하기 위한 화면과 쇼컷화면, 기대이미지 후보 리스트로 구성된다.

■ ImageViewer : 기대이미지를 편집하기 위한 화면으로 편집창과 쇼컷화면으로 구성되어 있다. 편집기능은 레이어 기능으로 여러 이미지를 조합하여 기대이미지를 만든다.

3.4 GUI 테스트 분석기

GUI 테스트 분석기는 GUI 테스트케이스의 기대이미지와 실행 결과 화면을 분석한다.

■ AnimatedImageEngine : 동적으로 움직이는 화면을 비교하기 위한 비교엔진이다. 동적 화면 비교는 여러 정지화면들의 리스트로 이루어지며 많은 비교 화면으로 테스트 시간에 영향을 준다. 옵션에 따라 비교 횟수를 조절하여 테스트 시간을 조절한다.

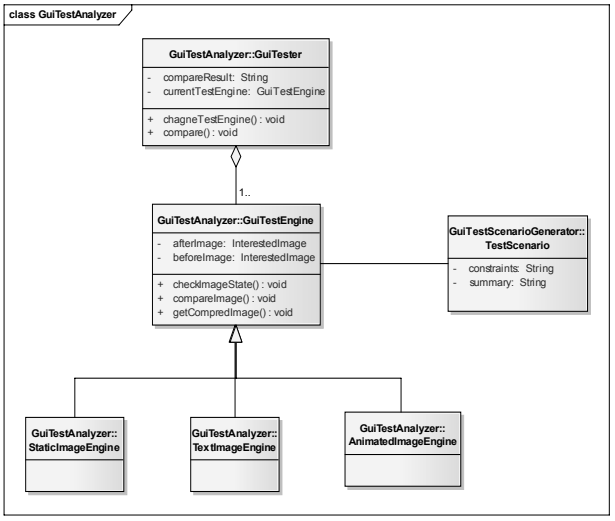
■ GuiTestEngine : 테스트엔진의 상위클래스. 화면의 형태에 따라 테스트엔진을 특화하여 테스트시간과 테스트성능을 향상시킨다. 이미지 수행 상태를 확인하고 2개의 이미지를 모은 후 비교한다.

■ GuiTester : GUI 분석을 테스트엔진을 관리하고 기대화면과 실제 화면을 비교한다.

■ StaticImageEngine : 일반적인 정적인 화면을 비교하기 위한 테스트엔진이다. 비트맵방식으로 이미지를 비교한다.

■ TextImageEngine : 화면 중에서 텍스트로 이루어진 부분을 비교하기 위한 비교 엔진이다. 텍스트의 내용을 추출하여 비교한다.

(그림 5)는 GUI 테스트 분석기 클래스 다이어그램이다.



(그림 5) GUI 테스트 분석기의 클래스 다이어그램

3.5 GUI 테스트 관리기

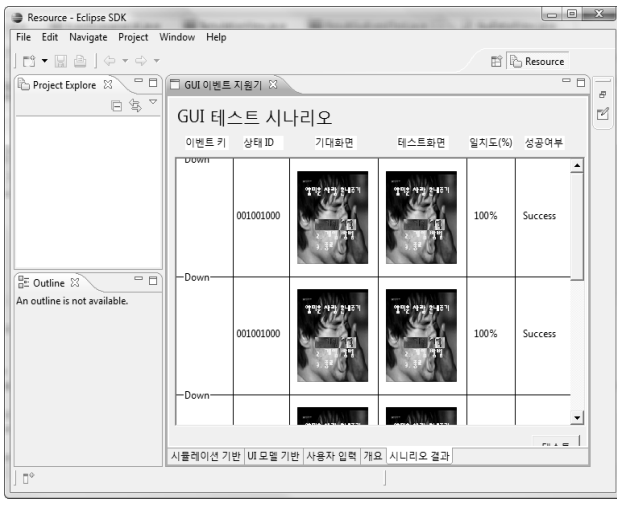
(그림 6)처럼 GUI 테스트케이스 관리기는 GUI 테스트 시나리오를 편집할 수 있는 화면을 제공하고 모바일 S/W에 이벤트를 보내고 실행 결과 화면을 수집한다.

또한 GUI 테스트 시나리오와 GUI 테스트케이스를 관리한다. 에뮬레이터 또는 모바일 디바이스에서 동작하는 모바일 S/W와 연결하여 S/W를 실행시키고 실행결과를 입력받는다.

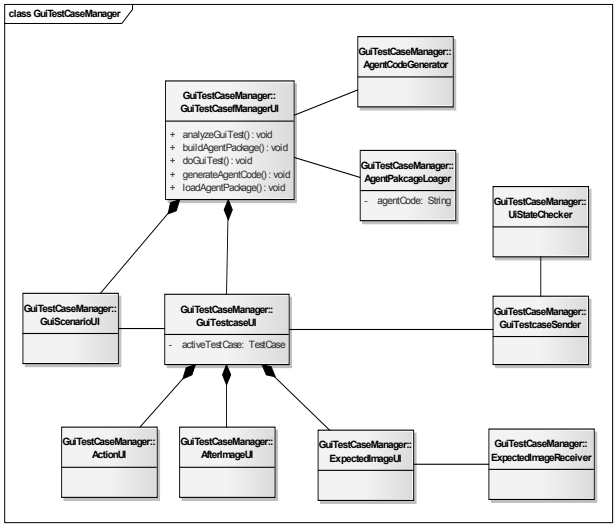
■ ActionUI : 데이터입력, 사용자의 액션을 추가하기 위한 화면을 제공한다.

■ AfterImageUI : 이벤트 발생 이후에 예상되는 화면을 등록하기 위한 인터페이스를 제공한다.

■ AgentCodeGenerator : 모바일 S/W에 이벤트를 보내고 실행결과 화면을 받기 위한 에이전트 코드를 생성한다.



(그림 6) GUI 테스트케이스 관리기 화면



(그림 7) GUI 테스트케이스 관리기의 클래스 다이어그램

■ AgentPakcageLoager : 에이전트 코드와 개발자가 개발한 모바일 S/W를 함께 빌드하여 에뮬레이터 또는 모바일디바이스에 탑재한다.

■ ExpectedImageReceiver : 이벤트 이후 모바일 S/W의 화면 결과를 전송받는다.

■ ExpectedImageUI : 이벤트 발생 이후 예상되는 화면을 편집하기 위한 인터페이스를 제공한다.

■ GuiScenarioUI : GUI 테스트를 위한 테스트 시나리오를 편집하기 위한 화면을 제공한다. GUI 테스트케이스 생성기에서는 테스트 시나리오를 위한 자료구조를 만든다. GuiScenarioUI 클래스는 자료구조를 읽어서 사용자가 사나리오를 편집할 수 있도록 한다.

■ GuiTestCasefManagerUI : GUI 테스트를 수행하기 위한 화면을 제공한다. GUI 테스트 시나리오를 보여주고 그에 따른 GUI 테스트케이스를 편집할 수 있는 화면을 제공한다. 실행 버튼을 누르면 자동으로 모바일 S/W와 연결하여 테스트 결과를 사용자에게 제공한다.

■ GuiTestcaseUI : GUI 테스트 시나리오에 포함되는 테스트케이스를 편집하기 위한 화면을 제공한다.

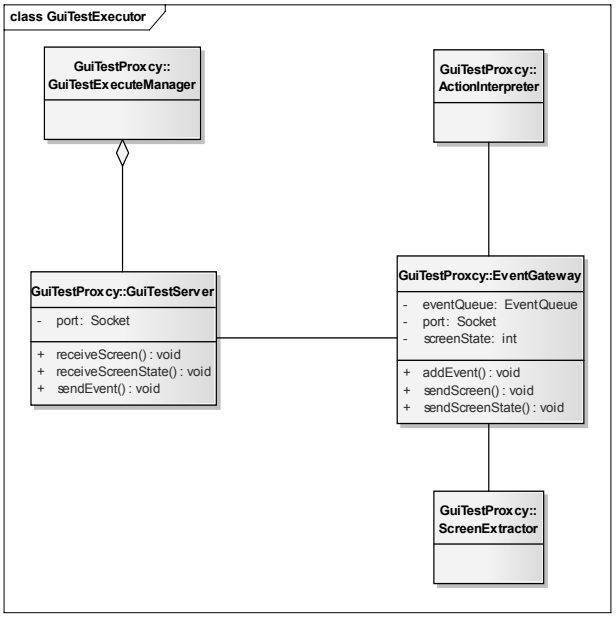
■ UiStateChecker : 이벤트 수행을 위한 조건을 확인하여 GUI 테스트 시나리오를 수행하는 도중 오류 상황이 발생할 경우 테스트를 중단한다.

3.6 GUI 테스트 에이전트

(그림 8)의 GUI 테스트 에이전트는 모바일 디바이스 또는 에뮬레이터와 GUI테스트 지원기에서 입력하는 이벤트를 입력받아 디바이스 또는 에뮬레이터로 전송하게 되며 또한 에뮬레이터에서 실행된 화면을 캡처하여 GUI 테스트 분석기로 전송하는 역할을 한다.

GuiTestCaseManager에서 보내는 이벤트를 받아서 모바일 S/W를 동작시킨다.

■ ActionInterpreter : Data 입력이나 사용자 액션을 해석하여 플랫폼에서 지원하는 이벤트로 변경한다.



(그림 8) GUI 테스트 에이전트의 클래스 다이어그램

■ EventGateway : Socket을 통해 GUI 테스트시나리오를 수행하기 위한 이벤트를 받고 실행 결과 화면을 보낸다. 데이터입력 등의 액션이 전달될 경우 액션을 분석하여 모바일플랫폼에서 사용하는 이벤트로 변경해준다.

■ GuiTestExecuteManager : GUI테스트지원기와 에뮬레이터 또는 모바일디바이스에서 동작하는 모바일 S/W의 연결부분을 관리한다. GUI테스트 시나리오에 따른 이벤트를 소켓을 통해 모바일 S/W에 보낸다.

■ GuiTestServer : Socket을 통해 모바일S/W에 이벤트와 실행화면을 주고받는다.

■ ScreenExtractor : 에뮬레이터 또는 모바일 기기에서 동작하는 S/W의 실행 결과를 수집한다.

4. GUI 테스트 주요 방법

GUI 테스트 지원기의 궁극적 목적은 모바일 응용 S/W의 GUI를 테스트 하는데 있다. 모바일 응용 S/W의 GUI를 테스트하기 위해서는 각각의 이미지마다 화면 ID, 키 이벤트, 키 이벤트에 따른 전후 이미지와 ID가 필요하다. 이러한 정보를 바탕으로 테스트 하고자 하는 GUI와 원본 이미지를 대조하여 자동으로 테스트 하게 된다.

4.1 이미지 상태 ID 정의 방법

GUI를 자동으로 테스트하기 위해서는 화면의 상태 정보를 알아야 한다. 모바일 응용 S/W는 다양한 변화가 있는데 크게 세 가지 변화로 구분된다.

4.1.1 전체 화면 변화에 따른 ID정의

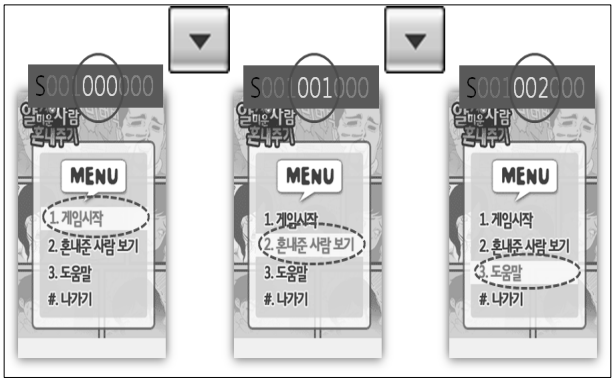
(그림 9)와 같이 전체 화면이 전환될 경우 처음 세자리가 순차적으로 변하게 된다. 첫 번째 화면의 ID가 'S001'이면

(그림 9) 전체 화면 전환시 ID 정의 방법

다음 화면은 'S002'로 정의 해준다. 이렇게 화면 ID를 정의하는 이유는 추후에 GUI를 비교하기 전에 화면 상태 정보를 먼저 비교함으로써 시간을 절약할 수 있는 장점이 있다.

4.1.2 이벤트 값에 의한 화면변화에 따른 ID정의

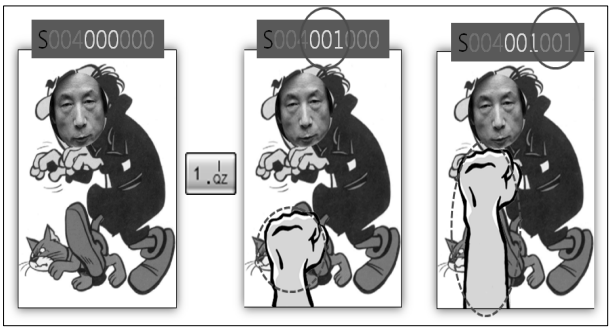
(그림 10)과 같이 전체 배경 화면은 같고 이벤트에 의해 부분적으로 화면이 변경될 경우 가운데 세자리가 순차적으로 변하게 된다. 첫 번째 화면의 ID가 'S001000000'이고 다음화면은 배경화면이 변하지 않았지만 메뉴화면만 변하였다. 이 화면의 ID는 'S001001000'로 정의 해준다. 추후에 이미지를 비교할 때 화면 ID 값을 먼저 비교하므로 이벤트를 비교하기 때문에 시간을 절약할 수 있다. 예로 화면 ID가 'S001001000'인데 테스트된 화면이 'S001002000'로 출력되었을 때 테스트 서버에서 화면 ID를 먼저 비교하여 오류를 감지하고 이벤트의 참거짓 유무는 판별하지 않게 된다.



(그림 10) 이벤트 값에 의한 화면 전환 시 ID 정의 방법

4.1.3 프레임 변환에 의한 화면변화에 따른 ID정의

(그림 11)와 같이 프레임에 따른 화면 변화는 일반화면 변화와는 차이점이 있다. 앞서 언급한 이벤트 값에 의한 화면 변환과 비슷하지만 한 개의 이벤트 값으로 화면이 여러 번 변하는 화면이다. (그림 10)에서 보는 것과 같이 처음 화면 상태는 'S004000000'이다. 이때 키 이벤트 값 '1'이 들어와 화면 상태는 'S004001000'이 되었다가 바로 'S004001001'의



(그림 11) 프레임별 화면 전환 시 ID 부여 방법

화면 상태로 변한다. 이것은 한 개의 이벤트 값이 들어와도 여러 화면이 변할 수 있다는 것이 된다. 따라서 이러한 화면의 테스트도 필요하기 때문에 화면 상태 ID를 정의 해주어야 한다.

4.2 화면 상태정보 생성방법

GUI를 자동으로 테스트하기 위해서는 각각의 이미지마다 정보를 가지고 있어야 하는데 이 정보는 XML 형식으로 저장한다. <표 2>는 각 이미지에 대한 XML 스키마이다. 한 화면에 대한 이벤트 타입과 이벤트 값, 그리고 전,후의 이미지 값을 가진다.

<표 2>에서 보면 화면 ID가 "S001001000"인 화면이 키 이벤트 'down' 값이 들어오면 화면 ID값이 'S001002000'인 다음 화면을 출력하고 'up'키를 누르면 'S001003000'인 화면을 출력하게 된다. 화면에서 동작하는 모든 키 이벤트 값을 정의해 주고 해당 이벤트 값에 따른 다음 화면 ID를 지정해 준다. 이벤트가 들어왔을 경우 원본 이미지 서버에서 해당 이미지를 자동으로 불러오게 된다.

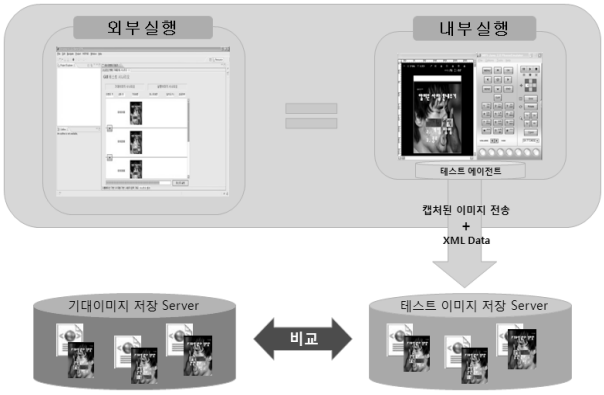
〈표 2〉 한 화면의 XML 스키마

```
<item>
  <itemid="S001001000">
    <event eventType="KeyClick" eventValue
     ="down", nextItem="S001002000"/>
    <event eventType="KeyClick" eventValue
     ="up", nextItem="S001003000"/>
    <event eventType="KeyClick" eventValue
     ="enter", nextItem="S002000000"/>
</item>
```

4.3 GUI 테스트 방법

GUI 테스트 시나리오 생성기에서 시나리오를 생성하면 시나리오에 포함된 각 화면의 정보들이 GUI 테스트 분석기를 통하여 테스트를 하게 된다. GUI 테스트 분석기는 각 화면에 포함된 XML 정보를 비교·분석하여 테스트를 하게 된다.

(그림 12)는 전체적인 GUI 테스트 흐름도이다. 테스터가 테스트 시나리오를 작성한 후 테스트 실행 버튼을 클릭하면 GUI 테스트 지원기는 테스트 시나리오 흐름대로 이벤트를 전송하여 테스트가 자동으로 시작된다. 에뮬레이터는 내부 실행으로 사용자의 눈에 보이지 않게 된다. 테스트 시나리



(그림 12) GUI 테스트 방법

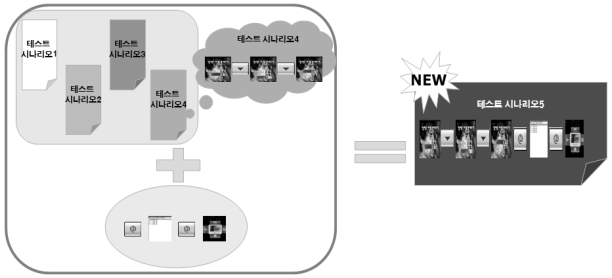
오의 화면 상태, 이미지, 이벤트 값 등의 정보가 테스트 에이전트를 통해 에뮬레이터로 들어온 후, 에뮬레이터는 정해진 시나리오의 순서에 따라 구동된다. 이때 에뮬레이터는 S/W를 동작시키는 동시에 테스트 에이전트를 통하여 테스트 이미지 서버에 수행된 이미지를 캡처하여 저장한다.

그리고 기대이미지 서버는 이미지와 XML 정보들을 바탕으로 비교 분석을 수행하게 된다.. 먼저 화면 상태를 비교한 후, 화면 상태가 정보가 일치하면 이미지를 비교한다. 이미지는 픽셀비교를 통해 정확하게 오류를 검출해 낼 수 있다.

4.4 테스트 시나리오 재사용 방법

사용자가 테스트를 하는데 있어서 기존의 테스트 시나리오를 재사용한다면 많은 시간과 자원이 절약될 수 있을 것이다. 이와 같이 GUI 테스트 지원기도 사용자가 기존의 테스트 시나리오를 이용하여 새로운 시나리오를 작성할 수 있다. 테스트 시나리오 재사용 방법은 아래 (그림 13)과 같다.

이 방법은 저장해 놓은 시나리오에 사용자 직접 입력 방법을 병행하는 것이다. 저장해 놓은 시나리오를 불러온 후 사용자가 새로 작성하고자 하는 테스트 시나리오에 맞게 이벤트와 이미지를 끼워 넣음으로써 재사용성을 확보하고 반복적인 작업을 탈피하는 장점이 있다.



(그림 13) 시나리오 재사용 방법

4.5 테스트 및 결과 및 화면 구성

4.5.1 실험 테스트

테스트 프로그램은 이번 실험을 위하여 제작한 간단한 프로그램을 대상으로 한다. '얄미운 사람 혼내주기'라는 모바일 응용 S/W로 핸드폰으로 찍은 사진이나 다운로드 받은 사진을 타깃으로 지정하여 간단한 키패드 조작으로 얄미운 사람을 혼내주는 프로그램이다.

4.5.1.1 실험 테스트 환경

가. Test Mobile Software

- 얄사혼(얄미운 사람 혼내주기)

나. 개발 환경

- OS : Windows Vista
- 개발환경 : Eclipse 3.4(GANYMEDE)
- 개발언어 : J2ME Java
- Emulator : .KTF-AROMA

다. 화면 구성

(그림 14) 얄사혼(Ver.KTF)의 화면 구성

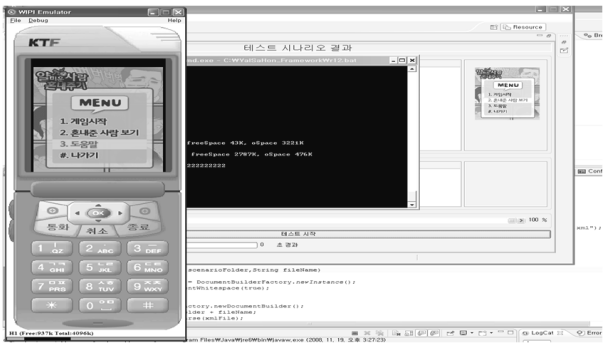
4.5.2 GUI 테스트 지원기 시나리오 생성 및 테스트 방법

GUI 테스트 지원기를 이용하여 테스트 자동화를 수행하기 위한 실행 화면이다. 테스터 및 사용자는 테스트 하고자 하는 시나리오를 구상한 후, 지원기 첫 화면의 추가 버튼을 클릭하면 시나리오를 작성할 수 있는 화면이 생성된다. 생성된 창에서 원하는 이미지와 이벤트의 순서를 입력한 후 (테스트케이스 작성) 테스트 단계로 넘어간다. 테스트 케이스는 시나리오에 기대이미지를 추가하여 테스트 수행 시 실제 화면과 비교 대상이 된다. 이 창에서는 시나리오의 수정, 편집, 삭제가 가능하다.

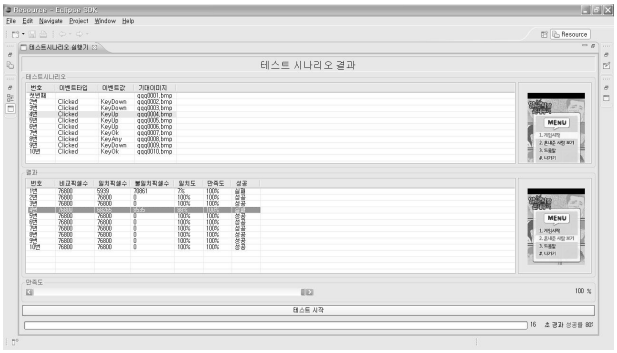
테스트 버튼을 누르면 (그림 16) 에뮬레이터를 구동시켜 사용자가 입력한 순서대로 테스트를 수행한다. 테스트 수행 후 결과를 (그림 17)와 같이 출력한다. 출력 결과는 실행될 때 캡처한 이미지, 비교 수치(%), 성공여부를 나타내 준다.

또 하나의 기능은 결과의 성공여부를 사용자가 임의대로 조절할 수 있다. 단순한 GUI는 100% 정확성을 요구할 수도

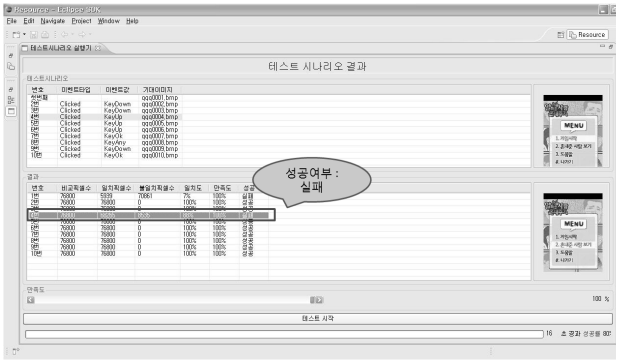
(그림 15) GUI 테스트 지원기의 UI



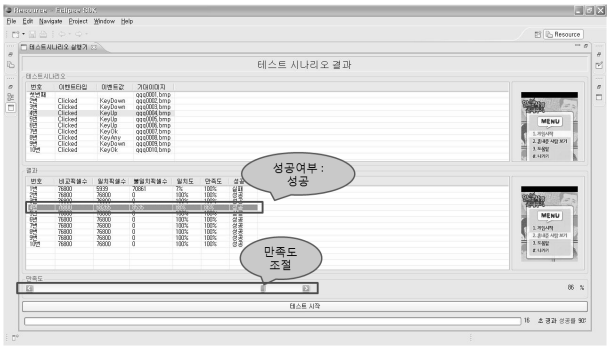
(그림 16) 테스트 수행 화면-에뮬레이터 구동



(그림 17) 테스트 수행 후 결과 화면



(그림 18) 만족도 조절 전



(그림 19) 만족도 조절 후

있지만 게임이나 복잡한 구조의 GUI는 사용자가 임의대로 정확성을 조절함으로써 사용자 만족도를 높일 수 있도록 하였다.

(그림 18)은 처음 결과 화면이다. 4번째 이미지가 실패이다. 옆의 이미지로도 확인이 가능하다. 하지만 사용자가 이 정도의 실패는 성공으로 하고 싶다고 한다면 (그림 19)처럼 만족도 조절 Bar로 성공여부를 변경 할 수 있다.

5. GUI 테스트 지원기 분석 및 평가

5.1 GUI 테스트 도구 비교

GUI 테스트 지원기와 기존 테스트 도구를 비교해 보았다. <표 3>에서 보는 것처럼 기능적인 부분을 개선하였고 또한 사용자가 손쉽게 사용하도록 UI를 수정 보완하였다.

또한 <표 4>에서 보는 것처럼 테스트 시나리오 5개와 그에 포함된 40개의 테스트 케이스를 수행 비교한 결과 기존 테스트도구 실행 시 80~90%의 검출율을 보였고 시간은 3분내외의 시간이 걸렸다.

반면에 GUI 테스트 지원기로 수행하였을 경우 100%의 결함을 발견하였을 뿐만 아니라 XML을 먼저 비교하기 때문에 상태 정보가 맞지 않으면 더 이상의 비교를 수행하지 않아서 테스트 수행시간이 기존 테스트 도구에 비해 1/3로 감소되었음을 볼 수 있다.

〈표 3〉 기존 도구와의 기능 비교

평가 항목	GUI 테스트 지원기	기존 테스트 도구
지원 플랫폼	다중플랫폼 지원 (SKVM, KTF-AROMA)	단일플랫폼 지원 (SKVM)
테스트 항목	- 이미지 일치도 검사 - 이벤트 일치도 검사 - 화면 상태정보(XML) 검사	- 이미지 일치도 검사 - 이벤트 일치도 검사
테스트 비교 방법	- 전체 픽셀비교 - 부분 픽셀비교	전체 픽셀비교
재사용성 여부	- 테스트케이스 저장 가능 - 테스트케이스 추가, 삭제 가능	- 테스트케이스 저장 기능 없음 - 테스트케이스 추가, 삭제 불가능
기대이미지 생성	가능	불가능
UI 가독성	좋음	나쁨

〈표 4〉 결함 검출율 및 수행 시간 비교

평가 항목	GUI 테스트 지원기	기존 테스트 도구
테스트 수	- 테스트 시나리오 수 : 5개 - 테스트 케이스 수 : 40개	
결함 검출율	100%	80~90%
테스트 수행시간 (동일 PC 사양)	1분 내외 (사용자의 PC사양에 따라 시간이 약간 다름)	3분 내외 (사용자의 PC사양에 따라 시간이 약간 다름)

5.2 GUI 테스트 지원기 장점 및 개선사항

앞 절에서 비교한 것과 같이 GUI 테스트 지원기는 작은 모바일 프로그램을 테스트 하는 도구이다. GUI 테스트 지원기의 가장 큰 장점은 사용자가 손쉽게 테스트를 할 수 있다는 점이다. 지원기의 간단한 절차를 통하여 테스트 하고자 하는 프로그램을 손쉽게 테스트 할 수 있다. 아래 <표 5>는 GUI 테스트 지원기의 장점 및 앞으로의 개선사항을 요약한 것이다.

향후 연구에는 현재 2개의 플랫폼을 사용하고 있어 제한적이지만 더 많은 종류의 플랫폼을 사용함으로써 더 다양한 테스트를 할 수 있는 환경을 조성해야 될 것이라고 생각된다. 또한 이러한 여러 개의 플랫폼을 실행할 경우 지금은 개별적으로 실행해야 하지만 이러한 환경을 하나의 환경으로 통합 한다면 사용자의 편의를 더욱 도모할 수 있을 거라 생각한다.

〈표 5〉 GUI 테스트 지원기의 장점 및 개선 사항

구 분	내 용
장 점	- 간단한 조작으로 자동화 테스트 가능 - 기존 테스트케이스 편집 가능 - 빠른 테스트 수행 시간 - 2개의 플랫폼 테스트 가능(SKVM, KTF) - 기대이미지 생성 가능 - 기존 도구보다 가독성 있는 UI 구현 - 상품화 될 경우 저렴한 비용으로 구매 가능
개선사항	- 다양한 플랫폼 기반의 테스트 연구 - 다중 플랫폼 테스팅 환경 통합

6. 결 론

현재 모바일 시장은 국내·해외를 막론하고 해가 거듭될수록 나날이 발전하고 있다. 모바일이 없는 세상은 이제 상상하기도 힘들 정도로 우리 일상생활의 한 부분을 차지하고 있다. 모바일 디바이스뿐만 아니라 거기에 탑재되는 모바일 응용 S/W 시장 또한 빠른 속도로 변화하고 있다. 모바일 응용 S/W의 성공 및 모바일 디바이스의 판매 실적의 가장 큰 영향을 미치는 요인이 바로 GUI이다. 구매자들은 GUI의 디자인을 보고 구매할 정도로 GUI가 미치는 영향은 기업 이윤에 막대한 영향을 미친다. 하지만 이러한 S/W의 GUI가 오류가 나거나 제대로 작동하지 않을 시에는 크나큰 손해를 감수해야 한다.

따라서 본 논문에서 이러한 문제점을 개선하고자 "모바일 응용 S/W GUI 자동화 테스트 방법 및 도구"를 제안하였다. 모바일 응용 S/W GUI를 간단한 방법으로 GUI의 오류 유무를 판별할 수 있으면 테스터 입장이 아닌 개발자 입장에서 테스트함으로써 구현 단계에서부터 오류를 줄일 수 있다. 또한 공개S/W인 이클립스(Eclipse)를 플랫폼으로 채택함으로써 대기업이 아닌 중소기업 및 벤처기업 등에서 부담없이 사용할 수 있다는 장점이 있다.

현재 GUI 테스트 지원기는 SKVM, KTF_AROMA등의 에뮬레이터에만 한정되어있고, 또한 터치폰 등의 GUI는 지원을 하지 못한다. 앞으로 모바일 응용 S/W GUI는 사용자와 교감할 수 있는 시스템으로 변환해 갈 것이다. 이것의 대표적인 예가 오늘의 터치 폰들을 들 수 있다. 사용자의 행동에 즉각적으로 반응하여 진동 등으로 표출함으로써 사용자가 교감을 느낄 수 있다.

향후에는 이러한 차세대 기능 및 다양한 플랫폼 등을 적용할 수 있도록 향후 연구가 필요할 것이라 사료된다.

참 고 문 헌

[1] LG경제연구원 전자 전략실 휴대폰담당 신동형, "휴대폰 산업 전망과 이슈", Oct., 2008.

[2] 윤석진, 김철홍, 신규상, "J2ME 기바 모바일 응용 S/W의 단위 테스트 도구 개발", 한국정보처리회 추계학술발표대회 논문집 제14권 제2호, 1033-1035, Nov., 2007.

[3] 채현철 ,이정주, 황선명, 정양재 "모바일 응용 S.W GUI의 자동화 테스트를위한 도구 구현", 한국정보처리학회 제15권 제1호, Nov., 2008.

[4] 채현철 ,황선명 "모바일 응용 S.W GUI의 자동화 테스트 및 관리를 위한 도구 설계 및 구현", 한국정보처리학회 제15권 제2호, May, 2008.

[5] 정일재, "시나리오기반의 모바일 응용 소프트웨어 GUI 테스트 방법에 관한 연구", 대전대학교 일반대학원 석사학위논문, Feb., 2008.

[6] 박상필, "이미지 플로우 기반의 모바일 GUI 테스트 도구에 관한 연구", 대전대학교 일반대학원 석사학위논문, Feb., 2008.

[7] 홍준성, "모바일 플랫폼의 기술현황 및 발전방향", 정보과학회지 제22권 21호 통권 제176호, pp.8-14, Jan., 2004.

[8] 한국정보통신기술협회, 소프트웨어테스트 전문기술 기초분야, 한국정보통신기술협회, 2005.

[9] 한국정보통신기술협회, 소프트웨어테스트 전문기술 응용분야, 한국정보통신기술협회, 2005.

[10] 김문겸, "모바일 게임GUI 디자인에 관한 연구", 청주대학교 대학원 석사학위논문, Dec., 2006.

[11] 채현철, 황선명, 김철홍"TDD기반의 모바일 단위 테스트 방법 및 개발에 관한 연구", 한국정보처리학회 추계발표대회 논문집 제14권 제2호, Nov., 2007.

[12] "MOPAD-I-MASTT 상세설계서" 한국전자통신연구원(2007-S032-01, 다중 플랫폼 지원 모바일 응용 S/W 개발환경 기술개발)

[13] 이상윤, 김선자, 김흥남, "한국 무선 인터넷 표준 플랫폼(WIPI)의 표준화 현황 및 발전 방향", 정보과학회지 제22권 21호 통권 제176호, pp.16-23, Jan., 2004.

[14] Sun-Myung Hwang, Hyeon-Cheol Chae, "Design & Implementation of Mibile GUI Testing Tool", ICHIT2008 International Conference, July, 2008.

황 선 명

e-mail : sunhwang@dju.ac.kr

1984년 중앙대학교 소프트웨어공학전공(이학석사)

1987년 중앙대학교 소프트웨어공학전공(이학박사)

2000년～현 재 한국 S/W프로세스심사인협회(KASPA) 이사

2000년～현 재 한국정보처리학회 논문지 편집위원

1997년～현 재 ISO/IEC J T C 7/WG10 한국운영위원

1989년～현 재 대전대학교 컴퓨터공학과 교수

관심분야 : 소프트웨어 프로세스 모델, 품질 메트릭스, 소프트웨어공학 표준화, 컴포넌트 품질측정, 테스팅 방법론 등