임베디드 소프트웨어 테스팅 이슈 및 현황

슈어소프트테크(주) 배현섭 · 윤광식 · 오승욱

1. 서 론

전자, 자동차 등 국내 주력 산업 전반에서 디지털 컨버전스 현상이 확연하게 진행되고 있다. 이에 따라 과거 하드웨어 서킷, 마이콤 등에 의해서 구현되던 임베디드 시스템들이 높아진 복잡도로 인하여 임베디드 소프트웨어로 대체 개발되는 현상이 확산되고 있다. 결과적으로 임베디드 소프트웨어의 품질 향상을 위한 테스팅에 대한 필요가 크게 증가하고 있다.

임베디드 소프트웨어 테스팅은 기능 테스팅, 신뢰성 테스팅, 성능 테스팅, 이식성 테스팅 등에 중점을 두고 수행된다는 점에서 일반 소프트웨어 테스팅과 유사점이 있다. 그러나 임베디드 소프트웨어는 일반 소프트웨어와 달리 개발 환경(호스트 환경)과 운용 환경(타겟 환경)이 다르므로 효과적인 테스팅을 위해서 기존 테스팅 도구를 개발 환경 및 운용 환경을 지원하기 위해서 변경, 확장할 필요가 있다[1,2,3]. 또한, 임베디드 소프트웨어는 테스팅 기술 측면에서 이벤트 방식의 테스팅 및 지속적인 피드백을 통한 테스팅 기법이 필요하다. 임베디드 소프트웨어들은 일반적으로 하드웨어 디바이스와 사용자로부터의 입력을 이벤트로 인식해서 처리한다. 또한 기능 수행을 위해서 다른 모듈이나 시스템과 지속적으로 피드백을 주고받으면서 동작한다. 따라서 함수 인터페이스나 사용자 인터페이스 위주로 진행되는 기존의 테스팅 기법을 이벤트와 피드백을 고려하여 수정해야 한다[7]. 따라서 본 논문에서는 **개발 환경(호스트 환경)과 운용 환경(타겟 환경)에 따른 테스팅 수행 방법, 그리고 이벤트 방식의 테스팅 및 피드백을 고려한 테스팅 기법에 대하여 정리하고 이에 대한 기술적 현황을 소개한다.**

이 논문의 구성은 다음과 같다. 2절에서는 임베디드 소프트웨어의 테스팅 환경 및 테스트 프로세스 관점에서 테스팅 도구의 확장 방안을 제시한다. 3절에서는 임베디드 소프트웨어의 특징과 이에 따른 테스팅 기법을 기술하고 마지막으로 4절에서는 결론 및 향후 발전 방향을 제시한다.

2. 임베디드 소프트웨어 테스팅 환경 및 프로세스

임베디드 소프트웨어는 일반적으로 호스트 환경이라 불리는 범용 개발 환경(Windows, UNIX, Linux 등)에서 개발되어, 타겟 환경이라 불리는 특정 임베디드 하드웨어 및 임베디드 운영체제 상에 탑재된다. 이 때 일반적으로 임베디드 소프트웨어는 그림 1과 같은 개발 및 테스팅 과정을 거친다[1].

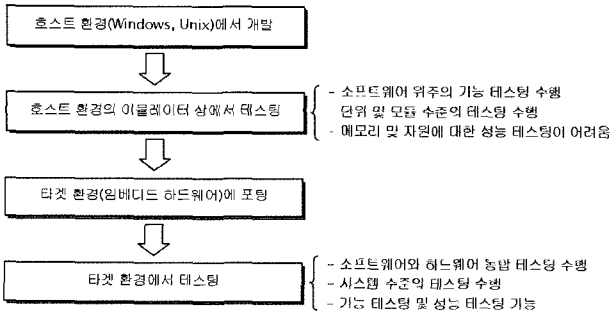

그림 1 임베디드 소프트웨어 개발 및 테스팅 과정

임베디드 시스템은 많은 경우에 하드웨어 및 소프트웨어 개발이 동시에 진행된다. 이는 하드웨어가 개발 완료되지 않은 상태에서 소프트웨어 테스팅이 필요한 경우가 많음을 의미한다. 이런 문제를 해결하기 위해서는 개발 호스트 환경에서의 테스팅 및 타겟 환경에서의 테스팅이 모두 필요하다[9]. 따라서 임베디드 테스팅 기술은 호스트 환경에서의 테스팅과 타겟 환경에서의 테스팅을 구분할 필요가 있다.

- **호스트 환경에서의 테스팅** : 호스트 환경에서의 테스팅은 임베디드 하드웨어에 대한 이뮬레이터(emulator)를 이용해서 이루어진다. 이뮬레이터는 타겟 환경과 유사한 테스팅 환경을 호스트에서 제공해 준다. 그러나 임베디드 디바이스 및 디바이스 드라이버, 메모리 용량, CPU 성능, 네트워

크 환경 등을 정확하게 이뮬레이션 하는 것은 매우 어렵다. 따라서 호스트 환경에서의 테스팅은 임베디드 소프트웨어 전체에 대한 시스템 테스팅 보다는 단위 테스팅 및 모듈 테스팅이 주로 수행된다. 또한, 시간 및 자원 사용에 대한 성능 테스팅보다는 기능 및 신뢰성 테스팅을 수행한다.

- **타겟 환경에서 테스팅 지원:** 임베디드 소프트웨어의 개발이 완료되고 탑재할 하드웨어 플랫폼도 확정되면 타겟 환경에서 테스팅이 수행된다. 임베디드 소프트웨어를 호스트 환경에서 충분히 테스팅 한 경우라도 임베디드 하드웨어가 확정된 후에는 타겟 환경에서 다시 한번 테스팅 할 필요가 있다. 이는 일반적으로 임베디드 시스템이 다양한 주변 기기를 포함하고 있는 경우가 많고 메모리 등의 하드웨어 자원에 제약이 있는 경우가 많은데 호스트 환경에서는 이 모든 상황을 완벽하게 이뮬레이션 하기가 어렵기 때문이다. 임베디드 시스템 타겟 환경은 일반적으로 메모리 등의 리소스에 대한 제약이 매우 심하므로, 테스트 도구 자체가 타겟 환경에 탑재되어 테스팅을 진행하는 것이 불가능하다. 따라서 테스트 도구는 호스트 환경에서 수행되고 타겟 환경에는 테스팅 진행을 도울 수 있는 테스트 에이전트(test agent)만 탑재되어 테스팅을 진행할 수 있도록 지원한다.

지금까지 기술한 임베디드 소프트웨어 테스팅 프로세스 및 환경을 고려해 볼 때 임베디드 소프트웨어 테스팅 도구는 테스트 케이스 설계 및 생성 모듈과 테스트 수행 모듈을 분리하는 구조가 바람직하다. 즉 테스트 케이스 설계 및 생성 모듈은 테스팅 수행 환경에 무관하게 재사용 가능한 테스트 케이스 및 테스트 프로그램을 개발할 수 있어야 한다. 이 때 테스트 케이스 설계 및 생성 모듈은 호스트 환경에서 동작하는 것이 일반적이다[1,9].

한편, 테스트 수행 모듈은 호스트 환경에서의 테스팅과 타겟 환경에서의 테스팅을 모두 지원할 수 있어야 한다. 호스트 환경에서 테스팅을 수행하는 경우에는 이뮬레이터와 연동하여 테스팅을 수행하고 결과를 모니터링 할 수 있어야 한다. 또한 타겟 환경에서 테스팅하는 경우에는 임베디드 운영체제와 연동하여 테스팅을 수행하고 결과를 모니터링 할 수 있어야 한다.

타겟 환경에서 수행되는 경우에 테스트 수행 모듈은 두가지 제약 조건을 가지게 되는데, 첫째는 메모리 제약 조건이다. 타겟 환경에서는 일반적으로 매우 한정된 양의 메모리만이 제공되므로 테스트 수행 모듈은 메모리 사용량을 최소화해야 한다. 둘째, 다양한 임베디드

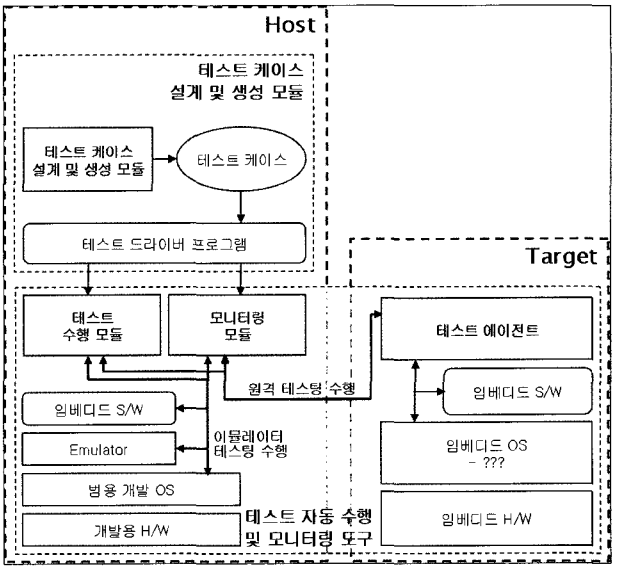

그림 2 임베디드 소프트웨어 테스팅 구조

운영체제와 쉽게 연동할 수 있어야 한다. 호스트 환경과 달리 임베디드 환경에는 매우 다양한 운영체제들이 사용되므로 운영체제의 변화에 따라서 쉽게 포팅 될 수 있는 구조를 가져야 한다[4,6].

그림 2는 지금까지 설명한 구조를 도식화 하고 있다. 그림 2에서 보는 바와 같이 테스팅 환경은 크게 호스트 환경과 타겟 환경으로 구분되며, 호스트 환경에서 동작하는 테스트 케이스 설계 및 생성 도구는 환경과 무관한 테스트 케이스와 테스트 드라이버 프로그램을 생성한다. 생성된 테스트 케이스와 테스트 드라이버 프로그램은 테스트 수행 모듈에 의해서 이뮬레이터 기반의 테스팅과 원격 테스팅에 모두 사용된다. 타겟 환경에서 테스팅을 수행하는 경우에 테스트 수행 모듈 전체가 타겟 환경에 포팅 된다면 메모리 사용량이 커지고 포팅이 어려워지는 문제점이 있다. 따라서 타겟 환경에는 테스트 에이전트만 포팅 되며 테스트 수행 모듈과 테스트 에이전트는 통신을 통해서 원격 테스팅을 진행한다. 이 구조에서 임베디드 소프트웨어 테스팅 도구는 크게 다음과 같은 세 부분으로 나뉜다.

- **테스트 케이스 설계 및 생성 모듈:** 임베디드 소프트웨어의 기능 및 성능 테스팅을 위한 테스트 케이스를 설계하고 생성한다. 테스트 케이스 생성은 테스트 프로시저 생성, 테스트 입력 데이터 생성, 테스트 예상 결과 생성 및 테스트 드라이버 프로그램 생성이 포함된다. 테스트 케이스 설계 및 생성 모듈은 호스트 환경에서 동작하며 일반적으로 타겟 환경과 무관하다.
- 이뮬레이터 테스팅 수행 모듈 : 호스트 환경에서의 테스팅은 임베디드 하드웨어에 대한 이뮬레이

터를 통해서 이루어지므로 이뮬레이터와 테스팅 도구 간의 인터페이스를 제공함으로써 호스트 환경에서의 기능 테스팅 자동화를 지원한다. 테스트 수행 모듈은 테스트 드라이버를 이용한 테스트 케이스 수행 모듈과 테스트 결과를 모니터링 하는 모듈로 나뉜다.

- **원격 테스팅 수행모듈:** 임베디드 시스템의 타겟 환경은 일반적으로 메모리 등의 리소스에 대한 제약이 매우 심하므로, 테스트 수행 모듈 자체가 타겟 환경에 탑재되어 테스팅을 진행하는 것이 불가능하다. 따라서 테스트 수행 모듈은 호스트 환경에서 수행되고 타겟 환경에는 테스팅 진행을 도울 수 있는 테스트 에이전트만 탑재되어 테스팅을 진행할 수 있도록 지원한다. 테스트 에이전트는 테스트 수행 모듈과 미리 정의된 프로토콜을 통해서 통신하며 테스트 케이스를 수행하고 수행 결과를 전달한다. 이와 같은 원격 수행 구조는 디버거에서 일반적으로 채택하는 원격 디버깅 구조와 유사하다.

3. 임베디드 소프트웨어 특징 및 테스팅 기법

임베디드 소프트웨어 테스팅은 범용 소프트웨어 테스팅에서 발생하는 모든 이슈를 다 포함하고 있을 뿐만 아니라 임베디드 환경 고유의 새로운 이슈들을 가지고 있다. 그림 3은 임베디드 소프트웨어 테스팅에서 발생하는 이슈들을 보여준다.

앞에 열거한 이슈 중에서 특히 이벤트 중심으로 동

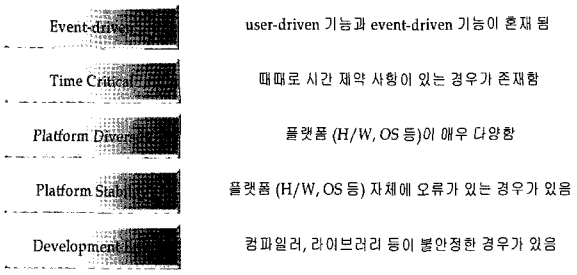

그림 3 임베디드 소프트웨어 테스팅 이슈

작하는 경우가 많다는 점이 임베디드 시스템의 특징이며 임베디드 시스템은 그 중에서도 비동기적 이벤트가 많기 때문에 테스팅 과정에서 이 부분에 대한 고려가 필요하다. 임베디드 소프트웨어의 기능 테스팅 기법으로 이벤트에 대한 캡처 및 재수행 기법(capture and replay)이 많이 사용된다[3,5]. 이 기법에서는 사용자 입력과 외부 메시지로 구성되는 이벤트에 대해서 임베디드 소프트웨어가 정확히 대응하는지를 테스팅 한다. 즉, 타겟 시스템에서 발생하는 사용자 입력 및 외부 이벤트를 녹화해서 테스트 스크립트를 구성하고 이 스크립트를 재수행 해서 결과를 확인하는 방법이다. 그림 4는 임베디드 소프트웨어에 대한 기능 테스팅 기법의 구성 및 흐름을 보여준다.

그림 4에 나타난 것처럼 기능 테스팅 자동화 도구는 테스팅 대상 소프트웨어가 탑재된 타겟 시스템과 타겟에 연결된 테스트 시스템으로 구성된다. 테스팅 관리자가 기능 테스팅을 위한 이벤트 녹화를 시작하면 단말기에서 발생하는 이벤트들이 테스트 에이전트에 의해서 검출되어 테스트 시스템에 전송되고 테스트 시스템

표 1 임베디드 소프트웨어 기능 테스팅 흐름

구분	단계	설명	비고
사용자 이벤트 녹화 (Record)	① 기능 테스팅 캡처 시작	테스팅 관리자 웹 브라우저를 통해 기능 테스팅 캡처 시작을 명령	
	②, ②' 이벤트 후킹 (event hooking)	타겟 시스템에 설치된 테스트 에이전트가 테스팅 대상 응용프로그램(IUT)에서 발생되는 이벤트를 가로챔	가로챈 이벤트는 XML로 가공되지 않은 RAW 데이터
	③ 이벤트 송신	타겟 시스템의 테스트 에이전트는 단말의 통신 인터페이스를 이용하여 테스트 시스템으로 이벤트를 전송	타겟 시스템에서는 통신이 가능한 네트워크 인터페이스를 제공해야 함
	④ 이벤트 저장	테스트 시스템의 캡처 모듈은 타겟 시스템의 테스트 에이전트가 송신한 이벤트 메시지를 XML 형태로 변경하여 저장함	수신한 RAW 이벤트 메시지를 XML로 변환하여 저장함
사용자 이벤트 재생 (Replay)	⑤ 기능 테스팅 재수행 시작	테스팅 관리자 웹 브라우저를 통해 기능 테스팅 재수행 시작을 명령	
	⑥ 이벤트 검색 및 변환	테스트 시스템의 재수행 모듈은 XML로 저장된 이벤트를 RAW 이벤트로 변환	RAW 이벤트로 변환
	⑦ 이벤트 통신전송	타겟 시스템의 테스트 에이전트에게 RAW 이벤트 메시지를 발송	
	⑧ 이벤트 전달	타겟 시스템의 테스트 에이전트는 수신한 RAW 이벤트 메시지를 IUT에게 전달	
	⑨, ⑨' 결과 전달	타겟 시스템의 테스트 에이전트는 테스팅 결과를 테스트 서버의 결과 처리 모듈에 전달	

에서는 검출된 이벤트를 스크립트 형태로 저장한다. 다시 테스팅 관리자가 저장된 스크립트를 선택해서 재수행을 시작하면 테스트 시스템에 존재하는 재수행 도구가 스크립트를 읽어서 단말기에 테스트 에이전트를 통해서 수행하고 결과를 피드백 받아서 저장한다. 표 1은 이와 같은 테스팅 흐름을 보여준다.

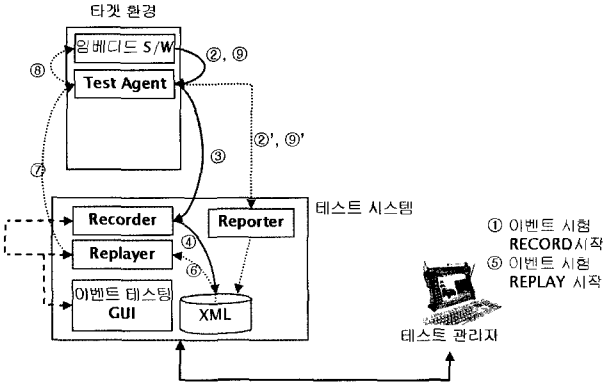

그림 4 이벤트 기반 기능 테스팅의 구성 및 흐름

지금까지 기술한 이벤트 기반의 테스팅 기법에 대한 적용 사례로서 모바일 소프트웨어 테스팅 사례를 고려할 수 있다. 휴대전화, 스마트 폰, PDA 등 모바일 분야는 현재 국내 산업계가 세계 시장을 선도해 가는 분야 중의 하나로서 임베디드 시스템 중에 가장 활발한 개발이 이루어지고 있다. 현재 휴대전화에 탑재되는 소프트웨어는 목적에 따라 개발 주체가 달라지며 이에 따라 품질관리 주체도 달라진다. 먼저, 휴대전화 단말기 제조사들은 폰을 동작시키기 위한 기본 소프트웨어를 휴대전화 개발 과정에서 탑재한다. 이에 포함되는 소프트웨어는 휴대전화용 운영체제, 하드웨어 추상화 계층(HAL, Hardware Abstraction Layer) 등이 포함된다. 기본 소프트웨어는 전화송수신 기능, 문자메

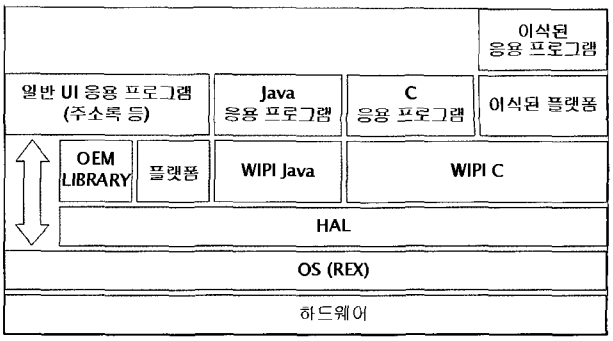

그림 5 휴대전화 소프트웨어의 구조

시지 송수신 기능을 포함한 휴대전화의 기본 기능을 제공한다.

기본 소프트웨어 계층 상단에 휴대전화용 기본 응용 프로그램이 탑재된다. 기본 응용 프로그램은 폰북 응용 프로그램, 일정 관리 프로그램 등 핸드폰 출시 시점에 이미 휴대전화에 내장된 응용 프로그램을 의미한다. 대부분의 경우에 휴대전화 응용 프로그램은 휴대전화 단말기 제조사에 의해서 개발되어 탑재된다. 이 때, 기본 응용 프로그램을 탑재하기에 앞서 별도의 플랫폼을 설치하고 이를 바탕으로 응용 프로그램을 개발하는 경우도 있고 운영체제 상에서 직접 응용 프로그램을 개발하는 경우도 있다.

마지막으로 휴대전화 출시 후에 다운로드 되어 동적으로 탑재되는 응용 프로그램들이 있다. 모바일 게임 등이 이에 해당하며 이 응용 프로그램들은 과거에는 통신 서비스 회사별로 별도로 정의된 플랫폼을 기반으로 개발되었다. 그러나 2005년 4월 이후로는 국내 출시되는 모든 휴대전화에 WIPI 플랫폼이 의무적으로 탑재되고 있으므로 조만간 휴대전화 응용 프로그램이 WIPI 플랫폼을 기반으로 통일될 것으로 예상된다.

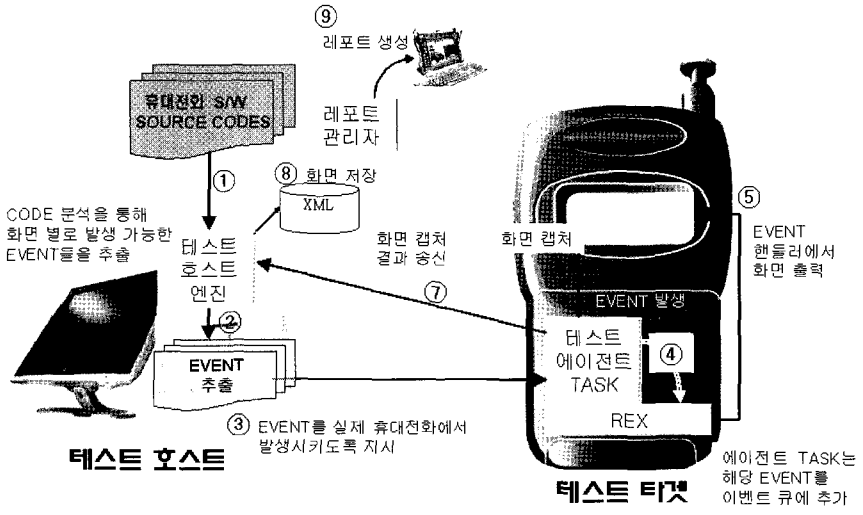

그림 6 휴대전화 소프트웨어 테스팅 환경 및 절차

그림 5는 이와 같은 휴대전화 소프트웨어 구조를 보여준다.

휴대전화 소프트웨어는 다른 여타의 임베디드 소프트웨어와 마찬가지로 이벤트 기반으로 동작한다. 따라서 휴대전화 자동화 도구는 이벤트 생성 및 리소스 모니터링에 중점을 두어야 한다. 그림 6은 휴대전화 소프트웨어 테스팅 자동화 도구의 구조와 이를 바탕으로 한 테스팅 흐름을 보여준다.

그림 6에서 보는 바와 같이 테스팅 자동화 도구는 테스트 호스트와 테스트 타겟으로 구성되며 휴대전화 소프트웨어에 대한 분석 결과를 바탕으로 이벤트를 생성시키고 이에 대한 휴대전화의 반응(화면 변화)을 캡처해서 결과를 판정한다. 이 과정에서 핵심적인 역할을 담당하는 테스트 에이전트는 휴대전화 운영체제인 REX에 포팅되어 운용된다.

이벤트 기반의 테스팅 이슈 외에 다른 이슈 중에서 시간 제약의 문제는 사실상 테스팅으로 해결하기가 거의 불가능하다. 테스팅 도구는 필연적으로 소프트웨어의 시간 특성에 영향을 끼치기 때문에 시간 제약이 핵심적인 경성 실시간 시스템(hard real-time systems)의 경우에는 테스팅 기법으로 품질 평가를 하기가 매우 어렵다[8]. 플랫폼의 다양성과 불안정성은 임베디드 시스템 테스팅을 어렵게 만드는 또 다른 요소이다[4,6]. 휴대전화, 디지털 TV, 자동차, 엘리베이터 등등 각 영역별로 서로 다른 하드웨어, 운영체제, 플랫폼을 사용하기 때문에 테스트 케이스의 재사용성 등이 떨어진다. 또한 임베디드 시스템에서는 운영체제나 디바이스 드라이버, 라이브러리를 수정하는 경우가 빈번하므로 응용 프로그램에 대한 테스팅뿐만 아니라 운영체제 수준에서의 테스팅 기법을 필요로 하는 경우가 많다.

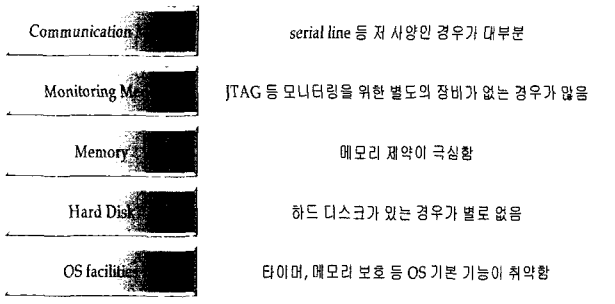

그림 7 임베디드 타겟 환경의 물리적 제약 조건

임베디드 소프트웨어 테스팅 도구의 경우에 타겟 환경에서의 테스팅을 지원하기 위해서는 전부는 아닐지라도 도구의 일부(2절에서 제시한 구조에 따르면 테스트 에이전트)는 필수적으로 임베디드 환경에서 동작해야 한다. 즉, 도구 자체가 임베디드 소프트웨어가 되는 셈인데 이때 테스트 에이전트는 그림 7에 명시된 물리적인 제약사항을 고려해서 개발되어야 한다. 구현에 있어서 특히 문제가 되는 부분은 저장 매체와 메모리의 부족이다. 테스팅 도구는 일반적으로 테스팅 과정에서 많은 양의 산출물을 발생하는데 이 산출물을 관리하기가 매우 어렵다. 게다가 호스트 환경과의 통신 환경도 시리얼 라인 밖에 없는 경우가 많아서 효율적인 데이터 관리가 매우 중요하다.

4. 결 론

임베디드 소프트웨어의 규모 및 복잡도의 증가에 따라 체계적인 테스팅에 대한 요구가 증가하고 있다. 임베디드 소프트웨어는 개발 환경과 운용 환경이 상이하므로 테스팅 환경도 이뮬레이션 기반의 테스팅 환경과 타겟에서의 테스팅 환경으로 구분된다. 이 논문에서는 테스트 케이스 설계 및 생성 모듈과 테스트 수행 모듈을 분리함으로써 이뮬레이션 환경과 타겟 환경에서 테스트 케이스를 공유하고 재사용할 수 있는 구조를 제시하였다. 또한 타겟 환경에서 원격 테스팅을 수행하는 경우에 타겟 시스템에 포팅되는 테스팅 모듈(테스트 에이전트)을 최소화함으로써 다양한 임베디드 운영체제 및 타겟 시스템에 대한 이식성을 높일 수 있는 원격 테스팅 구조를 소개하였다.

임베디드 시스템은 이벤트 기반으로 동작하는 경우가 많다. 사용자 입력뿐만 아니라 시스템 간의 상호작용이 모두 이벤트에 의해서 이루어지므로 테스팅 기법도 이벤트 기반의 테스팅을 반영하여야 한다. 이 논문에서는 이벤트 기반의 시스템을 테스팅하기 위해서 현재 가장 널리 사용되고 있는 캡처 후 재수행 (record and replay) 기법을 소개하였다. 이 기법에서는 임베디드 시스템 내에서 발생하는 이벤트를 테스트 에이전트를 통해서 캡처하여 스크립트로 만들고 이벤트 기반의 스크립트를 재수행함으로써 기능 테스팅을 수행한다. 또한 이와 같은 접근 방법을 핸드폰 상에서 동작하는 모바일 소프트웨어에 적용하는 사례를 설명하였다.

임베디드 시스템에서 자주 발생하는 실시간성과 인터럽트 처리에 대한 테스팅은 아직 해결되지 않고 있다. 테스팅 프로그램은 임베디드 소프트웨어의 시간 특성을 변화시키기 때문에 테스팅을 통해서 시간 오류를 발견하기는 매우 어렵다. 또한 임베디드 시스템에서 처리하는 인터럽트들은 일반적으로 비동기적으로 발생하는데 발생 시점에 따라서 시스템에 다른 영향을 끼치는 경우가 있다. 따라서 발생 시점을 고려해서 비동기적인 인터럽트가 시스템 기능 및 성능에 끼치는 영향을 테스팅하는 기법이 필요하다.

참고문헌

〔1〕 배현섭, "임베디드 테스팅 자동화 도구 및 요구사항", 임베디드 월드, Vol.31., pp.88-93, July 2005.

〔2〕 슈어소프트테크(주), 우수신기술 지정.지원사업 최종 보고서 - 임베디드 소프트웨어 테스팅 자동화 기술, 정보통신부, August 2004.

〔3〕 전자통신연구원, 정보통신 선도기반 기술개발사업 보고서 - 텔레매틱스 테스트베드 운영 기술 개발, 정보통신부, January 2006.

〔4〕 Remy Card, Eric Dumas, and Franck Mevel, The Linux Kernel Book, John Wiley & Sons, 1998.

〔5〕 Farnam Jahanian, "Run-Time Monitoring of Real-Time Systems," Advances in Real-Time Systems, pp.435-460, Prentice-Hall, 1995.

〔6〕 Jean Labrosse, MicroC/OS-II: The Real-Time Kernel, 2nd Ed., CMP, 1996.

〔7〕 Qing Li and Caroline Yao, Real-Time Concepts for Embedded Systems, CMP, 2003.

〔8〕 Jane Liu and Rhan Ha, "Efficient Methods of Validating Timing Constraints," Advances in Real-Time Systems, pp.199-224, Prentice-Hall, 1995.

〔9〕 Sol Shatz, Development of Distributed Software: Concepts and Tools, Macmillan Publishing Company, 1993.

배 현 섭

1993 KAIST 전산학과(학사)
1995 KAIST 전산학과(석사)
1999 KAIST 전산학과(박사)
2000 ETRI 컴퓨터소프트웨어연구소 선임연구원
현 재 슈어소프트테크(주) 대표이사
관심분야 : 소프트웨어 품질관리, 테스팅, 자동화, 임베디드 소프트웨어
E-mail : hsbae@suresofttech.com

윤 광 식

1995 KAIST 전산학과(학사)
1997 KAIST 전산학과(석사)
1999 KAIST 전산학과 박사수료
2001 MacroImpact(주) 연구소 책임연구원
2002~현재 슈어소프트테크(주) 첨단소프트웨어시험 센터장
관심분야 : 임베디드, 리엑티브, 실시간 소프트웨어 모델링 및 테스팅
E-mail : yoon@suresofttech.com

오 승 욱

1996 KAIST 전산학과(학사)
1998 KAIST 전산학과(석사)
2000 KAIST 전산학과 박사수료
2001 MacroImpact(주) 연구소 책임연구원
2002~현재 슈어소프트테크(주) 소프트웨어 시험 자동화 연구소장
관심분야 : 임베디드, 요구사항 검증 자동화, 테스팅, 자동화
E-mail : suoh@suresofttech.com