제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

타겟 기반의 임베디드 소프트웨어 테스팅 기술 및 시스템 개발에 관한 연구

김지혁*, 김상일*, 류성열**
숭실대학교 대학원 컴퓨터학과
e-mail : ji-hyeok@selab.ssu.ac.kr

A Study on Development for Target-Based Embedded Software Testing and System

Ji-Hyeok Kim, Sang-Il Kim, Sung-Yul Rhew
Dept. of Computing, Soongsil University

요 약

임베디드 시스템 개발과 임베디드 응용 소프트웨어 개발에 대한 관심이 높아지면서 이를 테스팅 하기 위한 기술과 테스팅 도구들의 도입이 확산되고 있다. 그러나 임베디드 시스템 테스팅은 여타 시험 검증 도구에 비하여 알고리즘의 정교성, 설계의 복잡성, 그리고 구현의 난이도 등으로 인하여 구현하기 쉽지 않은 것이 현실이다. 따라서 본 논문에서는 임베디드 시스템을 효율적으로 테스팅 하기 위해 타겟시스템 기반의 테스팅 방법을 제시하고, 테스팅을 위한 전체 시스템의 구성과 실시간 분석 방법을 제시한다. 이를 통해 임베디드 시스템을 테스팅 하기 위한 도구 구조를 분석하고 개발에 적용할 수 있도록 한다.

1. 서론

현재 임베디드 시스템을 테스팅 하기 위한 테스트 기술이나 시스템은 있지만 국내기술로 상용화된 임베디드 소프트웨어 시험검증 시스템은 없는 실정이다. 특히 학계나 산업계를 중심으로 시험 검증 방법 또는 기법 등에 관한 논문이 발표되고 있지만, 이들은 특정 임베디드 교차-개발환경에 한정되기 때문에 임베디드 산업 전 분야에 활용 가능한 상용 임베디드 소프트웨어 시험 검증 시스템은 없는 실정이다.[2] 따라서 본 논문에서는 "타겟 기반의 임베디드 소프트웨어 테스팅 및 실시간 분석 시스템"을 정의하는 기법을 제시하고자 한다.

본 논문에서 제안하는 "타겟 기반의 임베디드 소프트웨어 테스팅 및 실시간 분석 시스템"은 단말기, 핸드폰, 정보가전, 통신장비, 산업용 자동화 및 제어, 국방 및 항공, 자동차 및 교통, 물류 등의 다양한 산업에 사용되는 모바일 및 임베디드 소프트웨어 개발에 적용되는 기반 기술이다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구로서 임베디드 시스템 기술연구의 필요성과 임베디드 시스템 테스팅의 구성요소에 대해서 설명하고, 3 장에서는 임베디드 시스템을 테스팅 하기 위한 기법을 제시한다. 마지막으로 4 장에서는 결론을 내리고자 한다.

2. 관련연구

2.1 기술 연구의 필요성

기술 연구의 필요성은 중요성이 부각되고 있는 임베디드 소프트웨어의 특징과 산업적 가치에서 기인한다. 임베디드 소프트웨어는 일반 소프트웨어와 달리 실시간성, 고신뢰성, 저전력을 요구하는 특성을 가지며 통상적으로 개발 플랫폼과 타겟 플랫폼이 일치하지 않는다.[1] 이러한 개발환경을 "교차-개발환경(Cross Development Environment)" 이라고 부른다. 이러한 교차-개발환경을 구성하고 있는 타겟시스템은

매우 다양하고 복잡한 RTOS/Chip 벤더들의 조합으로 구성된다. 따라서 교차-개발환경을 기반으로 한 임베디드 소프트웨어 개발은 일반적인 데스크탑 컴퓨터에서의 개발방법론 및 도구와의 차이가 발생하게 한다. 따라서 고난도의 임베디드 소프트웨어 응용을 빠르고 안정되게 개발하기 위해서는 사용자가 쉽게 사용할 수 있는 기술이 절실히 필요하다.

그러나 보통 임베디드 시스템에서 제공하는 통합개발환경은 특정 교차 개발환경만을 지원하며 디버거 또는 하드웨어 기반의 Tracer 역시 풍부한 하드웨어 지식과 시스템 소프트웨어의 개발 경험을 요구하고 있으며 임베디드 소프트웨어의 시험 검증 시스템으로 사용하기에 큰 어려움이 따른다. 때문에 다양한 기종과 규격의 임베디드 소프트웨어 개발환경에 최적화된 별도의 시험 검증 기법을 연구해야 할 필요성이 절실하게 요구되고 있다.

본 논문에서 제시하고자 하는 기술의 구성은 다음과 같다. 첫째, 소프트웨어의 시험 검증을 생산적이고 효율적으로 적용할 수 있도록 하는 자동화 테스팅 기술. 둘째, 다양하고 복잡한 모바일 및 임베디드 소프트웨어 개발환경을 유연하게 대체할 수 있도록 하는 개방형 크로스-플랫폼 통합기술. 셋째, 타겟 기반의 시험 검증을 수행할 수 있도록 테스트 하니스(Test Harness)를 생성하는 소스코드 인스트러멘테이션 기술. 넷째, 실시간으로 시험 검증 결과를 확인하는 실시간 모니터링 기술. 다섯째, 타겟시스템의 오버헤드(Overhead)를 제어하는 실행 제어기술 등으로 구성된다.[3, 4]

2.2 기존 기술의 문제점 및 한계

종래의 기술의 문제점 및 한계 중 대표적인 문제점은 다음과 같다.

① 특정 또는 한정된 임베디드 교차-개발 환경에 적용 가능하거나 적용하려고 할 경우 상당한 작업 노력과 시간적 비용을 요구한다.
② 하드웨어 및 소프트웨어에 대한 고도의 지식을 필요로 한다.
③ 타겟시스템 기반의 시험이 불가능하다. 따라서 시뮬레이터 또는 에뮬레이터 기반의 테스팅을 수행할 수 밖에 없으면 정확한 소프트웨어 시험 검증을 수행할 수 없다.
④ 테스트 대상 소프트웨어의 실행을 실시간으로 분석하는 방법이 제공되지 않는다.
⑤ 획득된 결과를 이해· 활용하기 위해서는 별도로 고도의 분석 작업을 요구한다.
⑥ 테스트 케이스 및 테스트 모델이 불명확하다.
⑦ 시험 검증 프로세스가 쉽게 개발 프로세스에 흡수되지 못한다. 개발자의 추가적인 노력을 필요로 한다.
⑧ 임베디드 소프트웨어 시험 검증을 자동화된 기법으로 수행하지 못한다. 반복 테스팅 및 회귀 테스팅의 수행이 어렵다.
⑨ 상당한 투자비용을 필요로 한다.

3. 임베디드 소프트웨어 테스팅 및 실시간 분석 기법

이 장에서는 임베디드 소프트웨어를 테스팅 하기 위한 기법과 실시간으로 분석하기 위한 기법의 규칙들을 제시하고 각 구성요소에 대하여 기술한다.

3.1 임베디드 소프트웨어 테스팅 및 실시간 분석의 구성

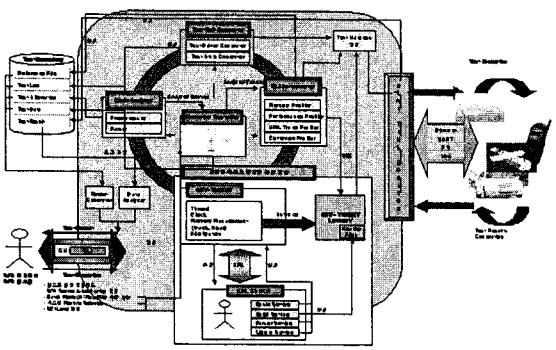
[표 1] 임베디드 소프트웨어 테스팅 및 실시간 분석을 위한 구성요소

기법	내 용
소스코드 문법 파싱 & Code Analyzer 모듈	각 컴파일러에 따라 작성된 소스 코드의 어휘와 문법을 완전히 분석하기 위한 파싱기술과 소스 코드 기반의 컴포넌트 테스팅과 실시간 분석을 위해 필요한 소스 코드 분석 모듈로 구성된다.
소스코드 삽입 기술 & CUI Instrumentor 모듈	해당 컴파일러가 완전하게 인식하는 새로운 소스코드 삽입 기술과 실시간 분석에 해당하는 4 가지 Testing Feature 를 적용하기 위한 Testing Library 삽입과 분석 Library 로 구성된다. 실시간 분석을 위한 Testing Feature - Memory Profiling - Runtime Trace Profiling - Performance Profiling - Code Coverage Profiling
개방형 크로스 플랫폼 적용 기술 & Any Target Library 모듈	개방형 크로스-플랫폼을 구현하기 위한 기술과 Testing Library 로 구성된다. 또한 다양한 임베디드 타겟시스템의 개발을 위해 필요한 제반사항 정의와 타겟시스템 실행을 위한 내부 System Call 에 대한 Library 사전을 정의한다.
컴포넌트 테스팅 기술 & Test Suite Generator 모듈	컴포넌트 테스팅의 Test Script 를 구현하기 위한 Test Script 언어 체계를 구현하고, 이를 기반으로 Test Driver 와 Test Stub 를 생성하는 모듈로 구성된다.
Execution Controller 모듈	Execution Controller 는 임베디드 시스템의 절차를 제어하고, 해당 동작에 대한 로그 정보를 남긴다. 그리고 실시간 분석 기능 선택 시 오버헤드에 대한 문제를 해결하기 위해 Instrumentation 의 Level 을 지정한다.
Real Time Monitor 모듈	해당 Target 으로 Test Harness 를 다운로드하고, Testing 수행 이후 테스트 결과를 해당 Target 개발 툴로 업로드 하는 것을 관리한다.
Data Analyzer 모듈	해당 Target 에서 생성되어 개발 툴로 업로드 되어진 하나의 통합된 Raw Test Result 를 각 Testing Feature 에 따라 분리하고, Report 생성에 적합한 형태로 변환 시킨다.
Report Generator 모듈	Data Analyzer 를 통해 생성된 각 Testing Feature 를 위한 Test Result 는 CUI Instrumentor 와 Test Suite Generator 를 통해 생성된 Reference File 과 조합하여 사용자가 확인 가능한 형태의 Report 를 생성한다.

3.2 임베디드 소프트웨어 테스팅 및 실시간 분석의 구축모형

본 논문에서 제안하는 "타겟 기반의 임베디드 소프트웨어 테스팅 및 실시간 분석 시스템" 구축모형의

구조는 [그림 1]과 같다.



[그림 1] 임베디드 소프트웨어 테스팅 및 실시간 분석 시스템 구축모형

[표 2] 실시간 분석을 위한 Testing Feature

기 능		내 용
컴포넌트 테스팅		C/C++ 언어의 Function, Procedure, Class 에 대해 자동화된 Unit, Integration Testing 기능을 제공한다.
실시간 분석	메모리 프로파일러	메모리 사용에 대한 통계와 Heap Memory Management 를 통하여 메모리 누수 검출 지점에 대한 프로파일링 기능을 제공한다.
	성능 프로파일러	소프트웨어 Code-Level 의 성능 데이터와 병목지점에 대한 프로파일링 기능을 제공한다.
	커버리지 프로파일러	소프트웨어 Code-Level 의 커버리지 기능을 제공하여 테스트 되어지지 않은 곳에 대한 프로파일링 기능을 제공한다.
	UML Trace 프로파일러	UML 을 이용하여 임베디드 소프트웨어에 실행에 대한 실시간 추적 기능 제공, 데이터 무결성 검사(Value Check) 제공한다.
해당기능별 모듈의 수행이 CLI(Command Line Interface)에 의해 완벽히 제어가능하기 때문에 수행목적에 따라 시스템을 세분화 할 수 있으며 개발업무프로세스에 시스템이 쉽게 적용가능 하다.		

[표 3] 적용시스템의 사양

구 분	내 용
지원 OS	· Windows 2000/XP Professional/Win2003 · RedHat Linux 7.0
지원언어	· C/C++ ※ ANSI C/C++를 기반의 다양한 임베디드 언어 지원
지원 통신채널	· Ethernet · File Systems · UART (Universal Asynchronous Receiver/Transmitter) · IrDA(Infrared Data Association)
지원 타겟	· 개방형 크로스-플랫폼 통합기술 채택으로 다양한 RTOS/Chip 벤더 지원 (8Bit RTOS ~ 64Bit RTOS) · 플랫폼 독립적인 시스템
사용자 인터페이스	· 직관적인 GUI(Graphic User Interface) · 세부화 되어진 CLI(Command Line Interface)
특징	· 낮은 메모리 범위와 한정된 타겟시스템 리소스 극복을 위한 다양한 SCI 수준(Level) 제어기능 제공 · (호스트)타겟시스템 기반의 실시간 분석기능 제공
	· 세분화된 사용자 Interface 의 제공으로 개발업무프로세스에 쉽게 적용가능 (Ex) 통합개발환경 또는 MakeFile 이용 · SCI 기술을 적용한 정형화된 리포트의 제공으로 개발 생산성 및 효율성 제고

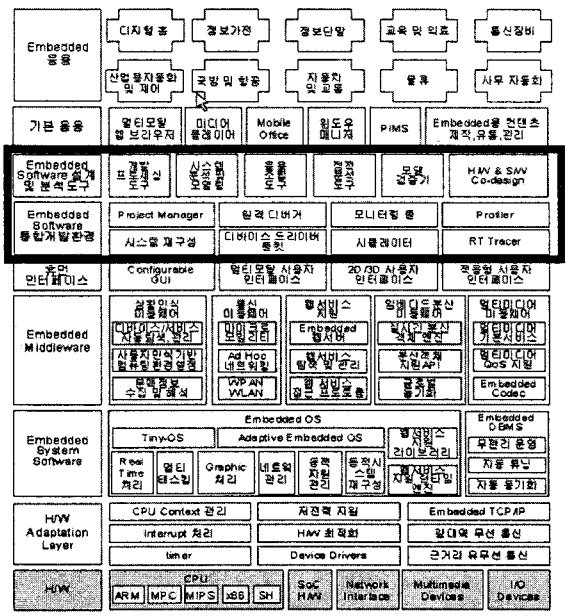
4. 결론

본 논문에서는 "임베디드 소프트웨어 테스팅 및 실시간 분석 시스템"을 위한 개발기법을 소개하였다. 또한 이 기법은 임베디드 시스템 개발 전 분야에 걸쳐 공통적이고 필수적인 핵심기술로서 [그림 2]와 같다.

이러한 기술적 소요를 만족하기 위해서는 사용자가 쉽게 사용할 수 있는 기술개발 및 시스템이 절실히 필요하다. 그러나 국외 상용개발업체가 소수이고 전통적인 통합개발환경 및 시험 검증 시스템 역시 특정 RTOS/Chip 벤더들만을 지원하는 경우가 대부분이다.

제한된 자원의 임베디드 시스템의 고신뢰성 확보를 체계적이고 효과적으로 달성하기 위해서는 다양하고 복잡한 임베디드 개발환경을 포괄하여 사용성과 효율성이 높은 시험 검증도구를 개발해야 한다.

따라서 본 논문에서 제시하는 개발기법은 복잡하고 어려운 임베디드 소프트웨어에 대한 응용을 빠르고 안정되게 개발하기 위하여 유용하게 사용될 수 있다.



[그림 2] 임베디드 소프트웨어 기능블록 구조도

참고문헌

[1] Arnold S. Berger, "Embedded Systems Design : An Introduction to Processes, Tools, and Techniques", CMP, 2001.

[2] Bart Broekman, Edwin Notenboom, "Testing Embedded Software", Addison Wesley, 2003

[3] Rational Corporation, "Rational TestRealtime Manual v2002.05.", 2002

[4] Kacsuk, P., "Performance visualization in the GRADE parallel programming environment", High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on, Volume: 1, 14-17 May 2000 Page(s): 446 -450 vol.1

[5] Krivosheev, O., McCrory, E., Michelotti, L., Mokhov, D., Mokhov, N., Ostiguy, J.-F., "A Lex-based MAD parser and its applications", Particle Accelerator Conference, 2001. PAC 2001. Proceedings of the 2001, Volume: 4, 18-22 June 2001 Page(s): 3036 -3038 vol.4

[6] Canfora, G., Cimitile, A., De Carlini, U., De Lucia, A., "An extensible system for source code analysis", Software Engineering, IEEE Transactions on, Volume: 24 Issue: 9, Sept. 1998 Page(s): 721 -740

[7] Subramanian, N., Chung, L., "Architecture-driven embedded systems adaptation for supporting vocabulary evolution", Principles of Software Evolution, 2000. Proceedings. International Symposium on, 1-2 Nov 2000 Page(s): 144 -153

[8] Fleischmann, J., Buchenrieder, K., "Prototyping networked embedded systems", Computer, Volume: 32 Issue: 2, Feb. 1999 Page(s): 116 -119

[9] Moore, L.J., Scarpino, F.A., "A debug technique for software on high performance processors an initial feasibility study", Aerospace and Electronics Conference, 1996. NAECON 1996., Proceedings of the IEEE 1996 National, Volume: 2, 20-23 May 1996 Page(s): 527 -534 vol.2

[10] Zipori, H., Sagiv, Z., Yossovitch, A., "Approaches and implementation of software test and development system for embedded computer systems", Computer Systems and Software Engineering, 1988. Proceedings., Third Israel Conference on, 6-7 June 1988 Page(s): 30 -39