KIPS Tr. Software and Data Eng.
Vol.4, No.9 pp.419~424 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2015.4.9.419

Development of Log-Based Testing Framework for Unit Testing of Embedded Software

Hodong Ryu† · Sooyong Jeong†† · Woo Jin Lee††† · Hwangsoo Kim††††

ABSTRACT

As Internet of Things (IoT) is recently serviced in several fields, the reliability and safety issues for IoT embedded systems are emerged. During the development of embedded systems, it is not easy to build the virtual execution environment and to test the developing version. Therefore, it is difficult to ensure its reliability due to lack of unit testing. In this paper, we propose a log-based unit testing framework for embedded software, which performs on real target board by extracting information of function execution. And, according to execution paths, duplicated logs are eliminated to keep a minimal log size. As a result, during system testing, testers can efficiently decide whether the executed paths of each function are correctly performed or not.

Keywords : Log-based Testing, Unit Testing, Embedded Software, Execution Path Analysis

임베디드 소프트웨어의 단위 테스팅을 위한 로그 기반 테스팅 프레임워크 개발

류 호 동† · 정 수 용†† · 이 우 진††† · 김 황 수††††

요 약

최근 사물인터넷 서비스가 점차 보편화되면서 사물인터넷에 사용되는 임베디드 시스템의 신뢰성 및 안전성 이슈가 부각되고 있다. 사물인터넷 시스템 등과 같은 임베디드 시스템 개발의 경우, 개발과정에서 소프트웨어 가상 수행환경을 구축하는 것이 쉽지 않아 단위테스트를 거의 수행하지 않고 있어 기능 신뢰성을 보장하기 어려운 실정이다. 이 연구에서는 실제 플랜트 환경에서 로그 기반 단위테스트 수행방법을 제안한다. 실제 시나리오 수행과정에서 함수별 수행경로를 고려하여 중복된 수행로그를 제거하여 저장되는 로그를 최소화하고 나중에 대표 수행경로만을 분석하여 단위테스트를 진행하는 방식이다. 제안된 방식은 시스템 테스트를 진행하는 과정에서 로그 기반 단위테스트를 효율적으로 수행할 수 있는 장점이 있다.

키워드 : 로그 기반 테스팅, 단위 테스팅, 임베디드 소프트웨어, 수행경로 분석

1. 서 론

최근 임베디드 시스템들을 연동하여 제공되는 사물인터넷 서비스들이 보편화됨에 따라 연계된 임베디드 소프트웨어의 신뢰성 및 안전성 이슈가 나오고 있다. 일반적으로 임베디드 시스템은 하드웨어와 소프트웨어 간의 높은 의존성을 지니고 있어, 소프트웨어의 수행을 위해서는 센서 및 액추에이터 등의 하드웨어를 통한 주변 환경과의 피드백이 필요하다[1]. 이러한 닫힌 루프(closed loop)의 피드백 환경이 제공되어야만 소프트웨어 수행이 가능하여, 임베디드 소프트웨어의 테스팅 기법에는 SW 가상 수행 환경 기반의 SiL (Software-in-the-Loop)[2] 또는 가상 프로토타입(Virtual Prototype, VP) 환경 기반의 VP 테스팅[3-4], 실물 프로토타입 기반의 HiL(Hardware-in-the-Loop) 테스팅[5], 실제 환경에서의 로그 기반 테스팅[6-8] 등이 있다.

※ 이 논문은 2012학년도 경북대학교 학술연구비에 의하여 연구되었으며, 미래창조과학부 및 정보통신기술진흥센터의 ICT융합고급인력과정지원사업 (IITP-2015-H8601-15-1002)과 정보통신·방송 연구개발사업의 일환으로 수행하였음[10041145, 자율군집을 지원하는 웰빙형 정보기기 내장 소프트웨어 플랫폼 개발].
† 준 회 원: LG전자 선임연구원
†† 준 회 원 : 경북대학교 컴퓨터학부 박사과정
††† 정 회 원 : 경북대학교 컴퓨터학부 교수
†††† 비 회 원 : 경북대학교 컴퓨터학부 교수
Manuscript Received : August 4, 2015
First Revision : August 28, 2015
Accepted : August 31, 2015
* Corresponding Author : Hwangsoo Kim(hsk@knu.ac.kr)

SiL, HiL, VP 기반의 테스팅 기법은 PC 또는 실물 HW 환경에서 가상 또는 실물 수행 환경을 구축하여 대상 소프트웨어를 수행하면서 테스트하는 기법으로 임베디드 소프트웨어의 초기 개발 단계부터 많이 사용되고 있는 대표적인 임베디드 소프트웨어 테스팅 기법이다. 하지만, 이러한 테스트 기법들의 적용이 쉽지 않은 이유는 가상 수행환경을 구축하는 것이 추가적인 비용과 시간이 소요되는 어려운 일이기 때문이다. 즉, VP 기반에서는 VP의 설계 및 구현, 에뮬레이션을 위한 테스트 환경 구축, HiL 기반의 테스팅에서는 하드웨어 기반의 테스팅 환경을 구축하여야 하므로 많은 부가적인 노력이 필요하다. 이에 반해 플랜트 기반 테스팅 기법의 하나인 로그 기반 테스팅 기법은 실제 수행환경에서 테스팅이 이루어지므로 가상 환경을 구축할 필요가 없으며 로그 생성에 필요한 최소한의 코드만을 대상 장치에 삽입하고 그로부터 생성되는 로그를 이용하여 장치의 수행과정 및 결과를 확인한다. 일반적으로 로그 기반 테스팅은 대상 장치에서 생성된 로그를 분석하여 의미 있는 행위를 추출하는 방식이다. 하지만 대상 장치에 추가된 프로브 코드(probe code)와 로그 저장 메모리 등의 리소스를 사용하여야 하므로 리소스가 부족한 임베디드 시스템에 적용하기에는 어려움이 따른다.

이 연구에서는 대상 시스템의 리소스를 최소로 활용하고 로그의 양도 최소화하면서 소프트웨어의 테스트 커버리지를 높일 수 있는 임베디드 소프트웨어의 단위테스트 방법을 제안한다. 일반적인 단위테스트는 함수별로 테스트 케이스를 만들어 진행하지만, 이 연구는 시스템 테스팅을 진행하면서 동시에 Fig. 1과 같이 함수별 수행 결과 로그를 분석하여 단위테스트를 추가적으로 수행하는 방식이다.

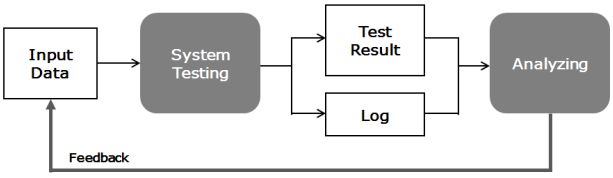


Fig. 1. Log-based Unit Testing Approach

먼저 각 함수의 호출과 그 과정에서 전달되는 인자 값 및 함수의 수행 후에 돌려받는 리턴 값을 로그로 남기는 기능을 가진 코드를 생성한다. 이때, 로그 생성 코드 삽입의 편의성을 위하여 관점지향 프로그래밍(Aspect Oriented Programming, AOP)의 AspectJ[9]를 이용하여 어드바이스(advice)로 테스트 대상 소프트웨어의 함수 호출과 리턴 부분에 해당하는 포인트 컷에 삽입한다. 전달된 로그는 PC 상에서 해당 함수의 수행 경로를 분석하여 이미 로그가 존재하는 경우는 해당로그를 버리고 새로운 경로의 로그만을 저장한다. 제시된 기법이 함수 단위로 이루어지므로 전역변수가 없는 모듈화 프로그램을 대상으로 한다. 전역변수가 포함된 경우는 함수별 분석이 전혀 의미가 없어진다. 본 연구에서 제시하는 도구는 추출하는 로그를 함수 단위로 제한하여 생성하고 저장되는 로그의 양을 감소시키고 함수별 단위 테스팅에 초점을 맞추어 수행 경로를 추출하여 수행 결과와 함께 테스터에게 제공한다.

본 논문의 구성은 다음과 같다. 먼저 2절에서는 임베디드 환경에서의 대표적인 테스트 기법과 기존의 로그 기반의 테스팅에 대하여 다룬다. 3절에서는 로그 기반의 테스팅의 전체적인 구조와 수행 과정에 대한 설명하고 프로브 코드 삽입방법과 PC 환경에서의 로그 분석 및 저장 방법을 설명한다. 4절에서는 사례연구로 본 논문에서 제시하는 기법의 효율성을 보이고, 마지막 5절에서 결론 및 향후 연구를 기술한다.

2. 관련 연구

임베디드 환경에서 각각의 소프트웨어 컴포넌트와 다른 소프트웨어, 하드웨어 컴포넌트 간의 높은 의존성은 대상 환경에서의 단위 테스팅을 어렵게 하는 주된 원인이 된다. SiL[2] 또는 VP 기반의 테스팅 기법[3-4], HiL[5] 등은 이러한 어려움을 극복하기 위한 대표적인 연구들이다. 하지만 이러한 기법들은 모두 고성능의 분석도구와 많은 노력과 비용이 필요해, 임베디드 소프트웨어 개발에 걸림돌이 되고 있다. 이에 반해 Plant 기반의 테스팅 기법인 로그 기반의 테스팅[6]은 Table 1과 같이 상대적으로 적은 노력만으로 테스팅을 수행할 수 있는 장점을 지니고 있으며 임베디드 환경에서의 디버깅 등에 널리 사용되고 있다.

Table 1. Comparison of Testing Approach for Embedded Software

Approach	Target Processor	HW Components
SiL based	SW Simulator	SW
HiL based	HW Simulator	HW + API
Plant based	Real Processor	Real HW

로그 파일을 이용한 분석방법[7]이나 테스트 오라클을 이용하여 로그로부터 수행 결과에서 오류를 분석하는 방법[8] 등은 로그로부터 수행과 관련된 정보를 추출하는 로그 기반 테스팅의 대표적인 사례이며, 이러한 연구들은 정형화되지 않은 로그 데이터로부터 의미 있는 정보를 추출하는 과정의 어려움을 보여준다. 누적된 로그는 그 자체로도 저장과 전달 관점에서 큰 부담이 되지만 이를 분석하는 일은 더욱 큰 노력을 필요로 한다. 이에 본 논문에서는 저장되는 로그의 양을 최소로 하면서 함수별 단위테스트를 수행하는 방법을 제안한다.

3. 로그 기반의 테스팅 프레임워크

본 논문에서 제시하는 로그 기반의 임베디드 소프트웨어 단위 테스팅 프레임워크는 Fig. 2와 같이 크게 타켓 임베디드 디바이스 부분과 PC 환경의 두 부분으로 나뉜다. 왼쪽

부분은 로그를 생성하는 대상 임베디드 장치이고, 오른쪽 부분은 이로부터 생성된 로그를 분석하여 해당되는 수행경로와 수행결과를 보이는 PC환경이다. 로그의 생성을 위해서는 먼저 대상 임베디드 장치에 로그를 남기는 프로브 코드의 삽입이 필요하다. 본 논문에서는 프로브 코드 삽입의 편의를 위하여 AOP 기법을 이용하여 함수 호출 및 리턴과 관련된 AspectJ의 포인트 컷에 로그를 저장하는 코드를 직조하는 방식을 사용한다[9]. 함수 호출이 일어나면 함수 호출 파라미터들과 리턴 시에 결과 값을 로그로 남기게 된다. 로그는 임베디드 시스템의 메모리 부담을 줄이기 위해 디버그용 시리얼 케이블을 통해 PC로 전달하여 저장한다.

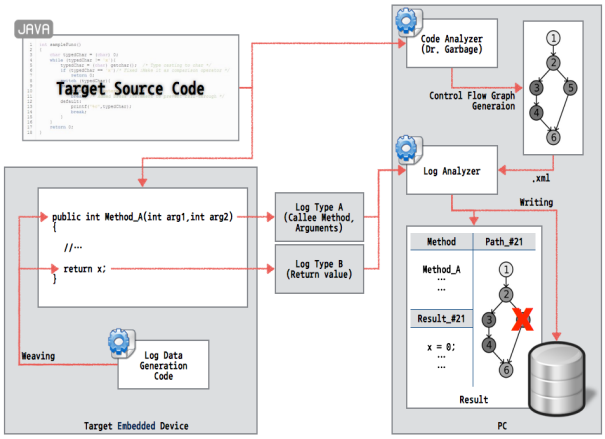


Fig. 2. Overview of Log-based Unit Testing Framework

PC 환경 중복 로그를 최소화하기 위해 해당 로그별로 수행흐름을 분석한다. 먼저 코드분석기(Code Analyzer)에서 대상 소스코드를 Dr. Garbage[10]를 이용하여 제어 흐름 그래프를 분석하여 패스 기본집합(a basis set of paths)[11]을 생성하여 XML 형태인 GraphXML[12] 정보로 변환한다. 로그분석기(Log Analyzer)는 생성된 로그정보와 제어흐름 그래프를 입력받아 함수수행 패스 정보를 파악하여 중복되지 않는 패스의 로그만을 저장한다.

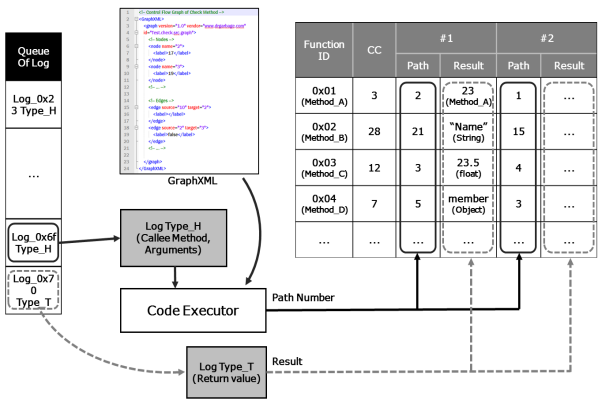


Fig. 3. Structure of Log Analyzer

Fig. 3은 로그 분석기의 내부 구조도를 보여준다. 로그큐로부터 로그가 전달되면 Code Executor가 GraphXML 정보를 이용하여 해당 로그의 수행 Path ID를 파악한 다음, 함수의 Path ID별로 로그를 기록한다. 이때 해당 Path ID에 이미 로그가 있으면 전달된 로그정보는 폐기한다.

일반적으로 최종 하위 함수의 경우는 위와 같은 방법으로 로그를 분석하면 되지만 다른 함수를 호출하는 호출 함수의 경우는 Path ID를 파악하는 것이 단순하지 않다. 로그 분석기의 목적은 Path ID를 파악하는 것이 목적이므로 호출되는 함수를 호출하는 대신, Fig. 4와 같이 stub 함수를 호출하여 stub 함수에서 로그파일로부터 해당 함수의 실제 수행 결과 값을 읽어오게 한다. 이렇게 하면 각 함수는 비록 호출관계가 있더라도 로그정보를 바탕으로 독립적으로 Path ID를 파악해낼 수 있다.

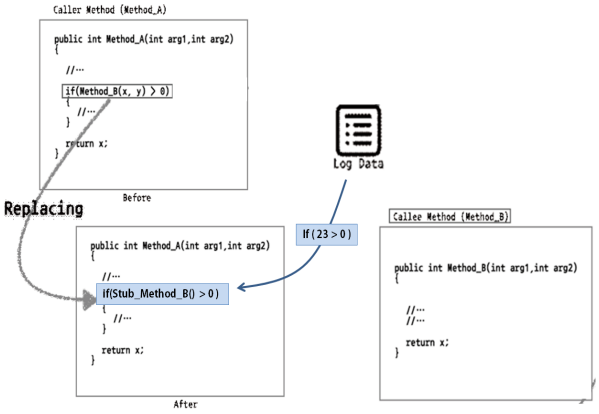


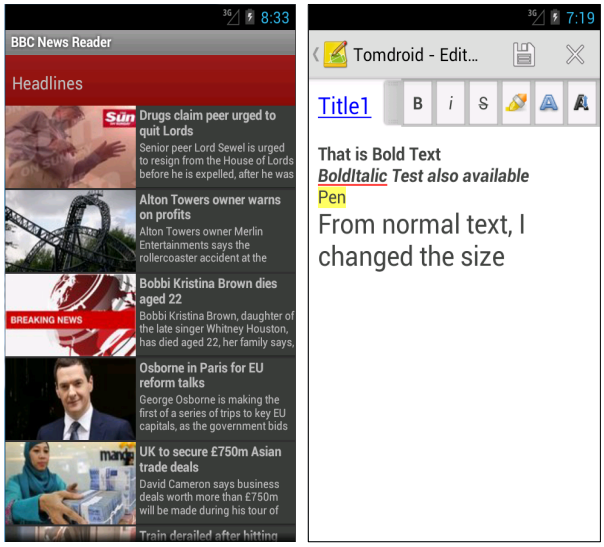
Fig. 4. Substitution Method of Calling Functions

Path ID를 파악하기 위해 Code Executor는 함수 내 각 분기문의 분기별로 고유 식별자를 부여하고 각 분기문의 어느 경로가 수행되었는지를 기록한다. 이후 수행된 분기의 식별자들만을 뽑아내고 1에서 '분기식별자 부분집합의 개수' -1 사이의 Path ID 값을 매핑 한다. 예를 들어, 함수 내에 IF 분기문이 2개 있는 경우에는 0-3번의 분기 식별자가 부여되며 Path ID 값은 1-15(24-1)의 하나로 매핑 된다. 순차적으로 IF문이 존재하고 참, 참 경로로 수행된 경우 수행된 분기 식별자 패턴은 (1, 0, 1, 0)이 되므로 Path ID 값은 10이 된다. 반복문이 있는 경우 반복수행 흐름은 아주 다양하게 존재하므로 이를 단순화하기 위해 분기 식별자 패턴을 단순 누적하는 형태로 진행한다. 반복문에서는 해당 분기문이 여러 번 수행되어 참/거짓 모두 수행될 수 있는데, 이러한 경우 분기식별자 패턴이 (1, 1, 1, 0) 형태가 될 수 있으며 이때 Path ID는 13이 된다.

4. 사례 연구

이 절에서는 기존의 함수 중심의 로그 방법과 제시된 패스 기반의 로그 방식을 비교함으로써 제안된 방식의 효용성

을 나타낸다. Fig. 5는 사례연구에 사용되는 안드로이드 공개 애플리케이션인 BBC 뉴스 리더(New Reader)[13]와 Tomdroid 메모[14]의 실행화면을 보여준다. 먼저 로그를 뽑아내기 위해 사용하여 Fig. 6과 같이 각 메소드의 호출이 있을 때에 해당 파라미터의 값을 저장하는 AspectJ의 Trace aspect를 사용하여 해당 앱의 소스 코드를 자동으로 변경하여 사용하였다.



(A) BBC News Reader (B) Tomdroid Memo

Fig. 5. Android Applications for Case Study

```
public aspect Trace {

	pointcut methodCalls():
		execution (* com.github..*(..))
		&& !within(com.github.Trace);
	...

	List<String> tracelog = new ArrayList<String>();

	before(): methodCalls() {
		String mStart = "Method Started => " +thisJoinPointStaticPart.getSignature().toString();
		System.out.println(mStart);
		tracelog.add(mStart);
		Object arguments[] = thisJoinPoint.getArgs();

		for(int i=0; i<arguments.length; i++) {
			Object argument = arguments[i];
			if(argument != null) {
				if(argument instanceof View) {
					String res = "Resource->"+((View) argument).getResources().
							getResourceEntryName(((View) argument).getId());
					System.out.println(res);
					tracelog.add(res);
				} else {
					String args = "Args: "+argument;
					tracelog.add(args);
					System.out.println(args);
				}
			}
		}
	}
```

Fig. 6. Part of AspectJ Code for Extracting Calling Information

메소드 로그 추출 코드가 심어진 BBC 뉴스 리더 앱의 주요 기능을 나타내는 테스트 시나리오들을 수행하였다. 먼저, 최근의 실행 시점에 로드 되어있던 기사들의 헤드라인을 확인한 다음, Reload 버튼을 눌러 최신 정보로 갱신한다. 여러 종류의 헤드라인 기사를 읽고 특정 카테고리 내로 들어가서 헤드라인에 노출되지 않던 다른 기사 읽기를 반복한다. Table 2는 BBC 뉴스 리더 앱의 테스트 시나리오를 수행하는 동안, 주요 메소드별로 제안 방법과 기존 함수별 로그 방법의 로그 횟수를 비교하여 보여주고 있다. 기존 함수 기반 로그 방식에서는 함수별로 수행로그를 모두 저장하는 방식이므로 함수 단위테스트 관점에서 모든 로그를 분석하여야 하는 단점이 있는 반면, 제안 방식에서는 중복 수행되는 패스의 경우는 로그를 남기지 않으므로 수행 패스별 대표 로그만을 분석하여 단위테스트를 효율적으로 진행할 수 있는 장점이 있다. Table 2에서 볼 수 있듯이 제안한 방법의 로그 감소율이 기존 대비 75.0~90.9%로 감소함을 알 수 있다. 표에 나타난 분기 커버리지는 기존 방법과 제안된 방법이 동일하게 적용되는 것으로 비교잣대로 쓰인 것이 아니라 수행된 테스트의 적절성을 보이기 위함이다. 분기 커버리지가 높다는 것은 함수 내부의 제어 흐름이 대부분 커버되어 단위테스트가 효과적이라는 의미이며, 반면 낮다는 것은 추가적인 테스트 케이스로 분기 커버리지를 높이는 노력이 필요하다는 것을 나타낸다. 로그 감소율이 높다는 것은 수행된 경로들이 대부분 중복이 되어 있음을 의미하며, 본 논문에서는 중복 수행경로의 로그 정보를 제거하여 효율적인 단위테스트를 수행할 수 있다.

Table 2. Comparison of Logs for BBC News Reader app

Method Name	Function-based	Proposed Method	Branch Coverage	Reduction Rate
ArticleActivity.IncomingHandler.handleMessage()	12	3	100.0%	75.0%
ArticleActivity.onCreate()	9	2	100.0%	77.8%
ReaderActivity.IncomingHandler.handleMessage()	88	8	80.0%	90.9%
ReaderActivity.displayCategoryItems()	30	3	77.8%	90.0%
ReaderActivity.setLastLoadTime()	13	3	46.7%	76.9%

두 번째 예제는 Tomdroid 메모 앱으로, 테스트 시나리오는 2개의 새로운 메모를 생성한 후, Bold, Italic 등의 다양한 글자 속성을 변경해가면서 새로운 메모를 작성하는 것이다. 작성되는 메모의 내용에는 이메일이나 전화번호와 같은 하이퍼링크를 유도하는 내용이 포함될 수도 있으며, 작성이 모두 끝나면 저장한다. Table 3은 Tomdroid 메모 앱을 수행한 경우 제안 방법과 기존 함수별 로그 방법의 로그 횟수를 비교하여 보여주고 있다. Table 3에서 볼 수 있듯이 제안한 방법의 로그 감소율은 기존방법 대비 0~94.9%로 나타났다. 표에서 EditNote.afterTextChanged() 메소드는 294번 호출되었지만 대부분 중복된 제어 흐름으로 수행된 것이어

서 제안된 방법에서는 15회의 로그만 저장되어 로그 감소율이 94.9%가 되며 저장된 로그는 78.6%의 분기 커버리지를 만족한다. 메소드 EditNote.changeTextSize()의 경우는 함수 수행 횟수가 3번에 모두 다른 경로로 수행되어 제안된 방법에서도 그대로 저장된 경우를 나타낸다.

Table 3. Comparison of Logs for Tomdroid Memo App.

Method Name	Function -Based	Proposed Method	Branch Coverage	Reduction Rate
ViewNote.showNote()	9	3	78.6%	66.7%
EditNote.afterTextChanged()	294	15	78.6%	94.9%
EditNote.toggleButtonOnClick()	8	1	15.0%	87.5%
EditNote.changeTextSize()	3	3	75.0%	0.0%
EditNote.saveNote()	3	1	25.0%	66.7%

5. 결 론

본 연구에서는 상대적으로 많은 노력이 필요한 임베디드 소프트웨어의 단위 테스팅을 위해 함수 단위의 로그를 기록하고 이로부터 수행 경로를 분석하여 단위 테스팅에 사용하는 로그 기반의 테스팅 기법을 제안하였다. 그리고 사례연구를 통해 기존의 방법보다 적은 양의 로그 데이터로 모든 경로와 그 결과를 확인할 수 있음을 보였다. 향후 연구에서는 이러한 효율성을 활용하여 분산 환경과 같이 대량의 장치들이 동시에 대량의 로그를 생성하는 상황에서 효과적으로 동작하는 로그 분석 기법에 대한 연구가 필요하다.

References

[1] Adams, Jack A, “A closed-loop theory of motor learning,” *Journal of motor behavior*, Vol.3, No.2, pp.111-150, 1971.

[2] Riley, Patrick F., and George F. Riley, “Next generation modeling III - agents: Spades - a distributed agent simulation environment with software-in-the-loop execution,” in *Proceedings of the 35th conference on Winter simulation: driving innovation*, 2003.

[3] Wang, G. Gary, “Definition and review of virtual prototyping,” *Journal of Computing and Information Science in engineering*, pp.232-236, 2002.

[4] Hodong Ryu, et al., “A Testing Technique based on Virtual Prototype for Embedded Software,” *Journal of Institute of Embedded Engineering of Korea*, Vol.9, No.6, pp.1-9, 2014.

[5] Short Michael and Michael J. Pont, “Hardware in the loop simulation of embedded automotive control system,” *Intelligent Transportation Systems*, 2005.

[6] Elyasov Alexander, “Log-based testing,” in *Proceedings of the 2012 International Conference on Software Engineering*, 2012.

[7] Andrews, James H., “Testing using log file analysis: tools, methods, and issues,” in *Proceedings of 13th IEEE International Conference on Automated Software Engineering*, 1998.

[8] Dan Tu, Rong Chen, Zhenjun Du, and Yaqing Liu, “A Method of Log File Analysis for Test Oracle,” in *Proceedings of Eighth International Conference on Embedded Computing Scalable Computing and Communications*, 2009.

[9] Laddad Ramnivas, “AspectJ in action: enterprise AOP with spring applications,” Manning Publications Co., 2009.

[10] Dr. Garbage, Control Flow Graph Factory [Internet], http://ww.drgarbage.com/control-flow-graph-factory.html.

[11] Arthur H. Watson and Thomas J. McCabe, “Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric,” NIST Special Publication 500-235, 1996.

[12] GraphML, The GraphML File Format [Internet], http://raphml.graphdrawing.org.

[13] BBC News Reader [Internet], https://github.com/oscarkey/BBC-News-Reader.

[14] Tomdroid [Internet], https://launchpad.net/tomdroid/stable.

류 호 동

e-mail : hodong@gmail.com

2007년 경북대학교 전자전기컴퓨터학부(학사)

2009년 경북대학교 컴퓨터학부(공학석사)

2015년 경북대학교 컴퓨터학부(공학박사)

2015년～현 재 LG전자 선임연구원

관심분야 : 모델 기반 테스팅, 분산환경 기반 테스팅 등

정 수 용

e-mail : kyo1363@naver.com

2012년 경북대학교 컴퓨터학부(학사)

2014년 경북대학교 컴퓨터학부(공학석사)

2014년～현 재 경북대학교 컴퓨터학부 박사과정

관심분야 : 가상 프로토타이핑, 시뮬레이션 기반 테스팅 등

이 우 진

e-mail : woojin@knu.ac.kr
1992년 경북대학교 컴퓨터과학과(학사)
1994년 KAIST 전산학과(공학석사)
1999년 KAIST 전산학과(공학박사)
1999년~2002년 ETRI 선임연구원
2002년~현 재 경북대학교 컴퓨터학부 교수
관심분야 : 임베디드 소프트웨어 테스팅, 임베디드 소프트웨어 개발환경 등

김 황 수

e-mail : hsk@knu.ac.kr
1975년 서울대학교 전기공학과(학사)
1982년 미시간대학교 EECS(석사)
1988년 미시간대학교 EECS(박사)
1989년~현 재 경북대학교 컴퓨터학부 교수
관심분야 : soft computing, 기계학습, embedded computing