ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 15-02-09

j.inst.Korean.electr.electron.eng.Vol.19,No.2,186~192,June 2015
http://dx.doi.org/10.7471/ikeee.2015.19.2.186

소프트웨어 품질향상을 위한 소스코드 기반의 테스트 케이스 자동 생성에 관한 연구

A Study on the Automatic Generation of Test Case Based on Source Code for Quality Improvement

손 웅 진*, 이 승 호*★

Ung-Jin Son*, Seung-Ho Lee*★

Abstract

This paper proposes an automatic generation technology of test case based on API in source code for software's quality improvement. The proposed technology is comprised of four processes which are analyzing source code by using the Doxygen open source tool, defining API specification by using analyzed results, creating test design, generating a test case by adapting Pairwise test technology. Analyzing source code by using the Doxygen open source tool is the phase in which API information in source code such as the API name, input parameter and return parameter are extracted. Defined API specification by using analyzed results is the phase where API informations, which is needed to generate test case, are defined as a form of database by SQLite database on the basis of extracted API information. Creating test design is the phase in which the scenario is designed in order to be composed as database by defining threshold of input and return parameters and setting limitations based on the defined API. Generating a test case by adapting Pairwise test technique is the phase where real test cases are created and changed into database by adapting Pairwise technique on the base of test design information. To evaluate the efficiency of proposed technology, the research was conducted by begin compared to specification based test case creation. The result shows wider test coverage which means the more cases were created in the similar duration of time. The reduction of manpower and time for developing products is expected by changing the process of quality improving in software developing from man-powered handwork system into automatic test case generation based on API of source code.

요약

본 논문에서는 소프트웨어 개발 과정에서 소프트웨어의 품질향상에 필요한 소스코드의 API를 기반으로 테스트 케이스를 자동으로 생성할 수 있는 기법을 제안한다. 제안된 기법은 Doxygen 오픈소스 툴을 이용한 소스코드 분석 과정, 분석된 결과를 이용한 API 사양 정의 과정, 테스트 디자인 생성 과정, Pairwise Test 기법을 적용한 테스트 케이스 생성 과정 등의 4가지 과정으로 구성된다. Doxygen 오픈소스 툴을 이용한 소스코드 분석 과정은 소스코드의 API 정보인 API 명, 입력 파라미터, 리턴 파라미터 정보 등을 추출하는 단계이다. 분석된 결과를 이용한 API 사양 정의 과정은 추출한 API 정보를 바탕으로 SQLite 데이터베이스를 이용하여 테스트 케이스 생성에 필요한 API 정보들을 데이터베이스화하여 정의하는 단계이다. 테스트 디자인 생성 과정은 정의된 API를 기반으로 입력 파라미터, 리턴 파라미터의 임계치 설정, 제약사항 설정 등을 통해 시나리오를 디자인하여 데이터베이스로 구성하는 단계이다. Pairwise Test 기법을 적용한 테스트 케이스 생성 과정은 테스트 디자인 정보를 바탕으로 Pairwise 조합 기법을 적용하여 실제 테스트 케이스를 생성하여 데이터베이스로 구성하는 단계이다. 제안된 기법의 효율성을 평가하기 위하여 기존의 명세서 기반의 테스트 케이스 생성 방법과 비교한 결과 비슷한 시간 내에 훨씬 더 많은 테스트 케이스가 생성되며, 명세 기반 기법으로는 불가능한 소스코드에 대한 기능 검증도 가능하여 소스코드내 결함 위치도 확인할 수 있다. 따라서 사람의 인력을 통한 수작업에 의존적으로 진행하였던 소프트웨어 개발 품질 향상 과정을 소스코드의 API를 기반으로 자동으로 테스트 케이스를 생성함으로써, 노동력 절감 및 제품 개발 시간 등을 단축 할 수 있으리라 기대된다.

Keywords: Pairwise, GUI Testing, Testcase, Information Technology, Embedded Test

* Department of Electronic Engineering, Hanbat National Univ, ujson@nate.com, 042-821-1423
★ Corresponding author
Department of Electronics&Control Engineering, Hanbat National University, shlee@cad.hanbat.ac.kr, 042-821-1137
Manuscript received May. 11, 2015; revised May. 25, 2015 ; accepted May. 27, 2015

Ⅰ. 서론

현대의 소프트웨어 역할은 기존의 컴퓨팅 영역을 탈피해 가전, 무선 단말기, 산업 기기 등 전방위로 확산되면서 SW 개발 공정은 물론 개발이 끝난 제품의 SW 결함을 사전에 진단하고 정상 동작여부를 시험하는 SW 테스팅 역할의 중요성이 점차 부각되고 있다. 또한, 소프트웨어 테스팅 기술은 응용 프로그램 또는 시스템의 동작과 성능・안정성이 요구하는 수준을 충족시키는지 확인하기 위해 "개발 전 과정에 걸쳐" 결함을 발견하는 행위로 현재 그 중요성이 부각되어 IT (Information Technology) 산업 전반에 걸쳐 적용되고 있다. 한편, 스마트폰, 태블릿PC, 클라우드 서비스, 빅데이터(Big Data), Social Computing, 사물 인터넷 등 전략 기술 분야에서도 SW 테스팅을 적용하여 사전에 결함을 발견하고 성능의 안정성을 확보하기 위해 많은 노력을 기울이고 있다.[1-2] NIST(National Institute of Standards and Technology)는 소프트웨어 결함이 미국 경제에 미치는 비용은 1년에 600억 달러에 달하며 이는 미국 GDP의 0.6%에 해당하는 엄청난 규모라는 분석 결과를 발표하였다. 그러나, 실제 비용 추정 모델에는 소프트웨어의 신뢰도 저하에 따른 부차적인 손실, 즉 브랜드 이미지 저하, 고객의 충성도 저하, 매출 및 시장점유율 하락 등은 고려대상이 아니기 때문에 실제 유형, 무형적 손실을 합하면 예상하기 힘든 경제적 손실이 발생하고 있음을 알 수 있다. 한편, 이 매몰 비용(sunk cost)중에서 220억 달러는 더 나은 소프트웨어 테스팅을 통해 줄일 수 있는 비용이라는 분석 결과도 포함하여 발표하였다. 한편, 소프트웨어 개발 과정에서 단계별로 빠른 결함 발견 및 결함 수정을 통한 소프트웨어 품질향상을 위해 소프트웨어 테스팅이 중요하다. 일반적인 테스트 케이스는 시스템에서 제공하는 기능 및 메뉴 등의 명세를 이용한 명세 기반의 테스트 케이스를 생성하게 된다. 명세 기반 테스트 케이스는 문서를 통해 사람이 직접 작성하여 생성하게 때문에 사람의 노동력 및 많은 시간이 소요된다.

따라서 본 논문에서는 이러한 명세 기반 테스트 케이스 생성 방법의 단점인 노동력 및 많은 소요 시간을 해결할 수 있는 소스코드의 API를 기반으로 테스트 케이스를 자동으로 생성할 수 있는 기법을 제안한다.

Ⅱ. 제안하는 방법

본 논문에서 제안하는 방법은 Doxygen 오픈소스 툴을 이용한 소스코드 분석 과정, 분석된 결과를 이용한 API 사양 정의 과정, 테스트 디자인 생성 과정, Pairwise Test 기법을 적용한 테스트 케이스 생성 과정 등의 4가지의 과정으로 구성되어진다.

1. Doxygen 오픈소스 툴을 이용한 소스코드 분석 과정

Doxygen 오픈소스 툴을 이용한 소스코드 분석은 소스코드에 포함되어 있는 API 명, 입력 파라메터, 리턴 파라미터, API 설명 등 각각의 API 들에 대한 정보를 분석하여 추출하는 과정이다. 그림 1은 Doxygen 오픈소스 툴을 이용한 소스코드 분석 블록도이다. Doxygen 오픈소스 툴은 툴 전체의 실행을 관리하는 doxygen main과 입력과 출력에 대한 설정을 하는 config 파일, 소스 분석 방법과 분석 룰을 정의하는 tag file로 구성된다.

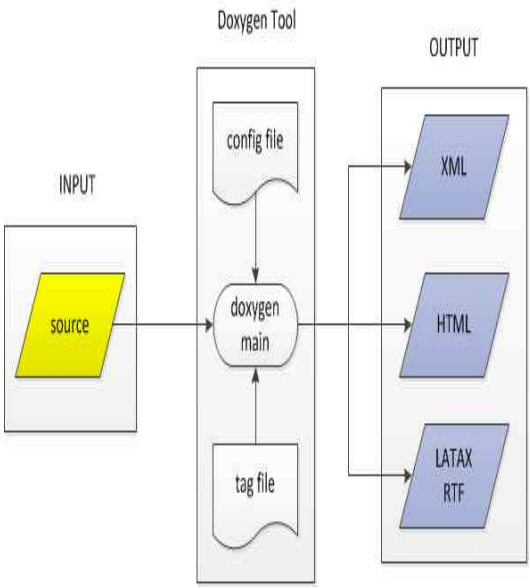

Fig 1. Analysis using Doxygen tool
그림 1. Doxygen 툴을 이용한 분석 과정

그림 2는 두수의 합을 구하는 샘플 프로그램이다. 이 프로그램을 사용하여 Doxygen을 통해 소스코드를 분석과정을 진행하였다. 그림 3은 추출한 결과를 사용자가 보기 쉽도록 html 파일 형태로 추출한 결과이다. 추출된 파일에는 소스코드의 API 명, 입력 파라미터, 리턴 파라미터, API 설명 등이 포함되어 있으므로 API 사양정의를 위한 기본 자료가 된다.

```
#include <stdio.h>

int sum(int i, int j)
{
/**
 *@brief Calculate the sum of the two integers..
 *@param input parameter i : Integer (Threshold 0 ~ 100)
 *@param input parameter j : Integer (Threshold 0 ~ 100)
 *@param return parameter rSum : Integer (Threshold 0 ~
200)
*/
    int rSum = i + j;
    return rSum;
}

/**
 *@brief Program to find the sum of the two numbers
 *@author Son Ung Jin
 *@date 2014/10/13
*/
void main(void)
{
    int nSum = 0;
    nSum = sum(10, 20);
    print ("SUM = %d", nSum);
}
```

Fig 2. Sample program to find the sum of the two numbers
그림 2. 두수의 합을 구하는 샘플 프로그램

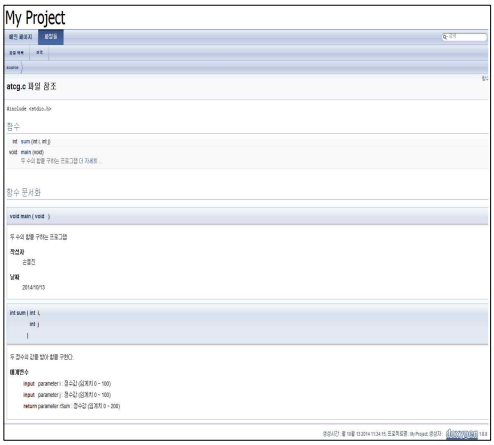

Fig 3. Result file of Doxygen
그림 3. Doxygen 결과 파일

2. 분석된 결과를 이용한 API 사양 정의 과정

분석된 결과를 이용한 API 사양 정의는 Doxygen 툴을 통해 추출된 소스코드의 API 정보들을 분석하여 API 명, 파라미터 이름, 입력값 등 API 사양을 정의하기 위해 필요한 정보들을 추출하는 과정이다. 추출한 데이터는 XML 또는 CSV, 데이터베이스 형태의 Repository에 저장할 수 있다. 본 논문에서는 SQLite라는 오픈소스 툴을 사용하여 데이터베이스화 하였다. 파라미터 정보는 입력 파라미터, 출력 파라미터, 리턴 파라미터로 구분되며, 각각의 파라미터는 데이터타입(int, char, user defined)을 가지게 된다. 입력 값의 경우 추출된 API 정보를 바탕으로 임계치를 설정할 수 있다. 그림 4는 그림 3에서 Doxygen 툴을 통해 추출된 API에 대한 정보를 토대로 API 사양을 API 명, 입력 파라미터, 출력 파라미터, 파라미터 값의 항목으로 구성한 후 데이터베이스화하여 테이블을 구성한 결과이다.

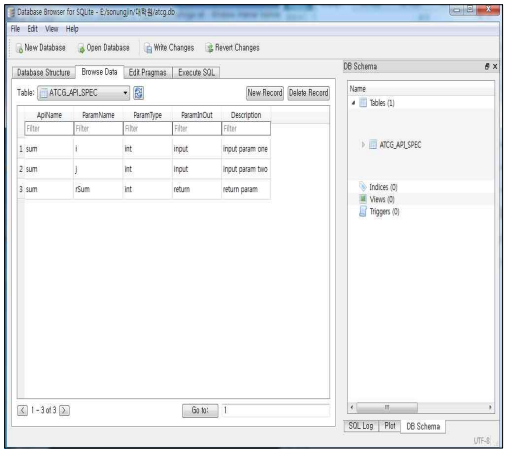

Fig 4. API Specification definition table
그림 4. API 사양 정의 테이블

3. 테스트 디자인 생성 과정

테스트 디자인 생성은 그림 4에서 정의된 API 사양을 바탕으로 테스트 디자인을 하는 과정이다. 테스트 디자인은 테스트 케이스를 작성함에 있어 기본이 되는 것으로 파라미터의 임계치 설정, 테스트 제약사항, 경험적 테스트 노하우 등의 정보를 바탕으로 어떠한 방식으로 테스트 케이스를 생성할 것인지에 대한 모델이다. API 사양 정의를 토대로 API에 대한 제약사항 설정, 임계치 설정, 필요한 또는 필요 없는 파라미터 쌍 설정 등을 통해 디자인이 가능하다. 실험에서는 숫자형 변수들만 존재하기 때문에 변수에 대한 임계치 설정만을 진행하였다. 그림 5는 파라미터의 임계치 설정 결과이다. 그림 6은 그림 5의 임계치 값을 이용하여 API의 각 파라미터의 테스트 디자인 생성하여 데이터베이스화 한 결과를 데이터베이스 브라우저로

확인한 결과이다. 파라미터가 2개이므로 행이 2개 생성된 것을 확인할 수 있다.

```
input parameter i -> 1 ~ 10
input parameter j -> 1 ~ 20
```

Fig 5. Threshold settings of each parameter
그림 5. 각 파라미터 임계치의 설정 값

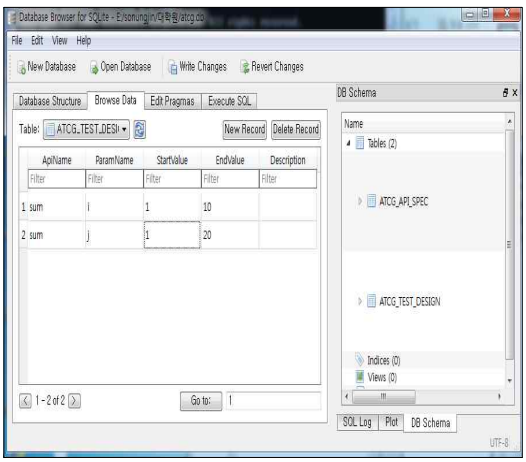

Fig 6. Output results of test design
그림 6. 테스트 디자인의 생성 결과

4. Pairwise Test 기법을 적용한 테스트 케이스 생성 과정

Pairwise Test[3-6] 기법을 적용한 테스트 케이스 생성은 그림 6에서 구성한 테스트 디자인과 Pairwise 테스트 기법을 이용하여 테스트 케이스를 생성하는 과정이다. 본 논문에서는 그림 6에서 생성한 2개의 파라미터의 테스트 디자인 정보를 이용하여 입력 값들을 Pairwise로 조합을 하면 그림 7과 같이 200개의 테스트 조합 결과가 생성된다. 입력 파라미터 i의 값이 1일 경우에 입력 파라미터 j의 값이 1부터 20까지 증가하며 20개의 테스트 케이스 조합이 생성되고, i의 값이 1씩 증가함에 따라 20개의 테스트 조합이 추가로 생성된다. 이와 같이 생성된 조합을 API 동작 기능에 맞게 2개의 파라미터 입력 값의 합을 구하도록 하고, 그 결과를 기대결과(Expected Result)라 한다. 또한 이렇게 생성된 테스트 조합을 데이터베이스화하면 그림 8과 같다. 각 파라미터의 값에 따라 2개의 파라미터 합이 기대결과가 된다. 테스트 조합이 200개이므로, 생성된 테스트 조합의 수도 200개가 된다.

i	j
1	14
1	20
1	12
1	1
1	4
1	19
1	7
1	9
1	2
1	6
1	15
1	10
1	18
1	13
1	8
1	11
1	16
1	5
1	3
1	17
2	11
2	2
2	3
.	.
.	.

Fig 7. Output results of test combination by Pairwise Testing technique

그림 7. Pairwise Testing 기법을 통한 테스트 조합 생성 결과

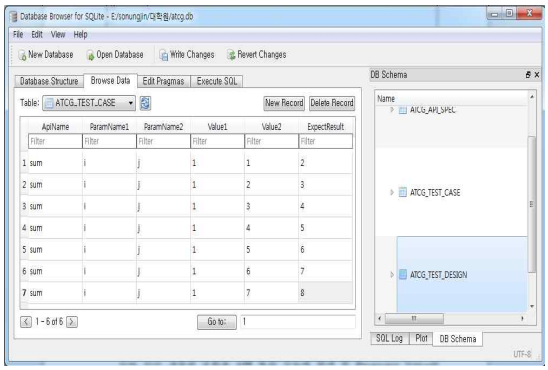

Fig 8. Output results of test combination
그림 8. 테스트 조합의 생성 결과

III. 실험 결과

1. 실험 환경

본 논문에서 테스트 케이스 생성 수를 측정하기 위하여 스마트 TV의 녹화 기능을 명세 기법을 통한 방법과 제안된 방법으로 테스트 케이스를 생성하여 비교 분석한다. 제안된 방법에 사용된 소스는 모사에서 개발한 스마트 TV의 소스 중 녹화 기능과 관련된 2000라인 이상의 소스코드 이용하였다. 이는 실제 스마트 TV의 녹화 기능에 사용되는 소스코드가 포함되며 녹화 관련하여 많은 API가 연계되어 동작되지만 실험을

위해 대표적인 API들을 이용해 실험을 진행 하였다. 스마트 TV의 녹화 기능은 사용자의 리모컨 조작 및 GUI 선택에 의한 방법으로 표 1과 같이 기능을 분류하여 선택할 수 있다.

2. 실험 결과

표 1은 스마트 TV의 녹화 기능이 3개인 경우를 나타낸 것이다. 화질은 일반 SD 화질과 HD 화질로 구분하고, 소리는 모노와 스테레오로 구분하고 저장매체는 내부 메모리와 외부 메모리로 구분하였으며 외부 메모리는 USB 메모리와 외장하드로 구분하였다.

Table 1. Smart TV recording function

표1.스마트TV녹화기능

Function	Parameter	
Video	SD	HD
Sound	Mono	Stereo
Memory	Internal Memory	External Memory
		USB / Sata2

(1) 명세 기반의 테스트 케이스 생성

표 1을 명세 기반의 직교배열 테스트 기법 중 각각의 인자 수에 따라 수준별 제곱 값으로 테스트 케이스를 생성하는 완전요인 배치 기법을 적용하여 테스트 케이스를 생성하면 표 3과 같이 12개($2^2 \times 3$)의 테스트 케이스가 생성된다.

Table 2. Test case outputs by orthogonal array

표 2. 직교배열을 통한 테스트 케이스 생성

No.	Parameter		
	Picture	Sound	Memory
1	SD	Mono	Internal Memory
2	SD	Mono	USB
3	SD	Mono	Sata2
4	SD	Stereo	Internal Memory
5	SD	Stereo	USB
6	SD	Stereo	Sata2
7	HD	Mono	Internal Memory
8	HD	Mono	USB
9	HD	Mono	Sata2
10	HD	Stereo	Internal Memory
11	HD	Stereo	USB
12	HD	Stereo	Sata2

(2) 제안된 기법을 통한 테스트 케이스 생성

표 1의 내용을 실제 구현된 소스코드를 통해 API 화하면 표 3과 같은 API를 구성할 수 있다. 표 4는 표 3의 API들에 대해 char형과 void형을 포함하면 거의 무한대의 테스트 케이스가 생성될 수 있기 때문에 int 형에 대해서만 임계치 설정한 결과이다.

Table 3. API of smart TV recording source code

표 3. 스마트 TV 녹화 소스코드의 API

API Name	Parameter
Input_GUI_Record()	int videoMode int soundMode int channelNum
deviceOpen()	int openMode int startPosition char recordFileName
recording()	int videoMode int soundMode int channelNum void reordingData
memory_Location()	int internal int external
external_Memory_Type()	int type int company int lockMode int fileSystem
check_Memory_Space()	int type int company int lockMode int fileSystem int checkMemorySpace (return value)

표 3의 API 들이 각 파라미터를 표 4의 임계치를 적용하여 pairwise 테스트 기법을 적용하여 테스트 케이스를 생성하면, 표 5와 같이 Input_GUI_Record() 함수의 테스트 케이스 생성 수는 20개, deviceOpen() 함수의 테스트 케이스 생성 수는 3개, recording() 함수의 테스트 케이스 생성 수는 20개, memory_Location() 함수의 테스트 케이스 생성 수는 4개, external_Memory_Type() 함수의 테스트 케이스 생성 수는 56개, check_Memory_Space() 함수의 테스트 케이스 생성 수는 56개로 총 159개의 테스트 케이스가

Table 4. Threshold and setting value

표 4. 임계치 및 설정 값

Parameter	Threshold
int videoMode	0: SD 1: HD
int soundMode	0: Mono 1: Stereo
int channelNum	1 ~ 10
int openMode	0 : Read Mode 1 : Write Mode 2 : Read/Write
int startPosition	Recording 0
int internal	0 : Unaware 1 : Awareness
int external	0 : Unaware 1 : Awareness
int type	0 : USB 1 : External Hard Disk
int company	0 : S Company 1 : L Company 2 : C Company 3 : T Company 4 : M Company 5 : U Company 6 : O Company 7 : A Company
int lockMode	0 : unlock 1 : lock
int fileSystem	0 : FAT32 1 : NTFS

생성되었다.

Table 5. Test case outputs per API entry

표 5. API 항목별 테스트 케이스 생성 수

API Items	Test-Case Num.
Input_GUI_Record()	20 unit
deviceOpen()	3 unit
recording()	20 unit
memory_Location()	4 unit
external_Memory_Type()	56 unit
check_Memory_Space()	56 unit

(3) 고찰

명세 기반 테스트 케이스 생성 실험과 제안된 기법으로 통한 테스트 케이스 생성 실험 결과는 표 6과 같다. 표 6에 나타난 바와 같이 명세 기반 테스트 케이스 생성 수는 12개인 반면에, 제안된 기법으로 테스트 케이스를 생성한 결과는 159개이다. 이 결과는 명세 기반 테스트 케이스 생성과 비교하여 13배 이상 많은 테스트 케이스가 생성됨이 확인되었다. 따라서 명세 기반의 경우는 소스코드 내부 결함에 대한 검증이 이루어지지 않는 반면에, 제안된 기법은 소스코드를 분석하고 소스코드의 API 별로 테스트 케이스가 생성되어 소스코드 내부에 잠재된 결함까지 검출 할 수가 있었다.

Table 6. Comparison of specification-based technique and proposed technique

표 6. 명세 기반 기법과 제안한 기법 결과 비교

	Specification-based techniques	Proposed techniques
Test-Case Num.	12	159
Source code for internal defect detection	Disable	Disable

Ⅳ. 결론

본 논문에서는 소프트웨어 개발 과정에서 소프트웨어의 품질향상을 위한 소스코드의 API를 기반으로 테스트 케이스를 자동으로 생성할 수 있는 기법을 제안하였다. 제안된 소스코드 기반의 테스트 케이스 자동 생성 기법은 소스코드를 Doxygen 오픈소스 툴을 이용하여 분석한 후, API 정보인 API 명, 입력 파라미터,

리턴 파라미터 정보 등을 추출하여 API 사양을 정의한 후 데이터베이스화 하였다. 데이터베이스화 된 API 사양은 각각의 파라미터에 임계치 설정을 통해 테스트 디자인을 생성하여 데이터베이스화 한 후, 데이터베이스화 되어 있는 테스트 디자인 정보를 토대로 Pairwise Test 기법을 적용하여 각각의 API 별로 테스트 케이스를 생성하였다. 실험결과, 기능이 3개일 경우에는 명세 기반 기법을 통해 생성된 테스트 케이스 수는 12개인 반면에 제안된 기법을 통한 테스트 케이스 생성 수는 159개로 약 13배 이상 많은 테스트 케이스가 생성되었다. 또한 제안된 기법은 기존의 명세기반 기법으로는 불가능한 소스코드의 API 기반의 검증이 가능하기 때문에, 소스코드 내에 잠재되어 있는 결함도 확인할 수 있었다. 따라서 본 제안 기법은 기존의 명세 기반 기법을 사람의 인력을 통해 수작업에 의존적으로 진행하였던 소프트웨어 개발 품질 향상 과정을 소스코드의 API를 기반으로 자동으로 테스트 케이스를 생성함으로써, 노동력 절감 및 제품 개발 시간 등을 단축 할 수 있으리라 기대된다. 한편, 앞으로의 연구에서는 소스코드의 API를 기반으로 테스트 케이스 자동 생성뿐 아니라 생성된 테스트 케이스를 이용한 테스트 자동화 방안의 연구가 필요하다.

References

[1] Q. Xie and A.M. Memon, "Model-Based Testing of Community-Driven Open-Source GUI Applications," Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on, pp. 145-154, Sep. 2006.

[2] J. H. Kim, "Android and Android Market", Journal of Contents Association, Vol.7, No.2, pp.29-36, 2009.

[3] Malte Lochau, Sebastian Oster, Ursula Goltz, Andy Schürr, "Model-based pairwise testing for feature interaction coverage in software product line engineering," Software Quality Journal Vol. 20, No. 3, pp.567-604, Sep. 2012.

[4] Amitava Mukherjee, "A Near-Nonparametric Partially Sequential Test for Monitoring Phase II Location Under Pairwise Dependence Between Two Phases," Sequential Analysis Vol. 30, No.2, pp.208-228, Feb. 2011.

[5] James Bach, "Risk and Requirements-Based Testing," IEEE Computer, Vol.32, No.6, pp.113-114, June, 1999.

[6] Luc Duchateau, Paul Janssen, "Pairwise non parametric non inferiority tests in 3 × 3 cross over trials: should we adjust for period," Statistics in Medicine Vol. 24, No.10, pp.1525-1536, May. 2005.

BIOGRAPHY

Ung-Jin Son Student Member)

2002 : BS degree in Electronic Engineering, Hanbat National University
2015 : MS degree in Electronic Engineering, Hanbat National University

Seung-Ho Lee (Member)

1986 : BS degree in Electronic Engineering, Hanyang University
1989 : MS degree in Electronic Engineering, Hanyang University
1994 : Ph. D degree in Electronic Engineering, Hanyang University
1994 ~ current : Professor,
epartment of Electronics&Control Engineering, Hanbat National University