제29권 제2호(통권106호) 2020년 11월 ISSN 1738-6322

# 소프트웨어공학소사이어티 논문지

## Journal of Software Engineering Society

VOLUME 29, NUMBER 2, November 2020



한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

SOFTWARE ENGINEERING SOCIETY

# 효율적인 소프트웨어 제품라인 회귀시험을 위한 자동화된 코드 기반 시험 방법†

(Efficient Code-based Software Product Line Regression Testing)

정 필 수‡ 강 성 원§

(Pilsu Jung) (Sungwon Kang)

**요 약** 소프트웨어 제품라인 개발은 제품군의 개발을 위하여 공통적인 부분과 가변적인 부분을 분리 개발함으로써 중복개발을 피하여 효율적으로 제품군을 개발하는 개발 패러다임이다. 소프트웨어 제품라인 개발에서 제품군을 생성하기 위해 사용되는 소스코드를 제품라인 코드 베이스라고 부르고, 제품라인 코드 베이스가 변경되어 제품군의 제품들이 영향을 받을 때 영향 받은 제품들을 시험하는 활동을 제품라인 회귀시험이라고 한다. 이 때 제품군의 각 제품을 개별적으로 시험하는 대신, 변경과 무관한 시험을 파악하여 피할 수 있다면 효율적인 제품라인 회귀시험이 가능해 질 것이다. 본 논문은 이런 방법으로 소프트웨어 제품라인 회귀시험을 효율적으로 수행하는 자동화된 방법인 SRTS를 소개한다. 이 방법은, 먼저 제품라인 코드 베이스와 시험 항목을 공통성과 가변성을 기반으로 나누고 변경에 영향을 받는 시험 항목을 식별하여 선택한 후, 선택된 시험 항목만을 재실행함으로써 불필요한 시험을 줄인다.

키워드: 소프트웨어 제품라인, 제품군, 회귀시험

**Abstract** Software product line development is a development paradigm that efficiently develops a product family by avoiding redundant development based on separation of the common part and the variable part of the product family. In software product line development, the source code that is used to produce a product family is called a product line code base, and when the product line code base is changed and the products of the product family are affected by the change, the activity of testing the affected products is called a product line regression testing. For product line regression testing, instead of conducting regression testing individually on each product of the product family, a more efficient regression testing would be possible if unnecessary testing that are irrelevant to the change can be avoided. This paper introduces SRTS, which is an automated method to efficiently perform software product line regression testing. SRTS divides the product line code base and test cases based on commonality and variability. Then SRTS identifies and selects the test cases affected by the change. Finally, it reduces unnecessary testing by rerunning only the selected test cases.

Key words: Software Product Line, Product Family, Regression Testing

## 1. 서론

소프트웨어 회귀시험은 소프트웨어가 변경되었을 때 변경된 부분을 재시험하여 이전에 시험된 코드에 결함이 없도록 검증하는 활동이다[1]. 이 활동은 소프트웨어가 변경될 때마다 반복적으로 수행되어야 하므로 전체 시험 비용에 큰 비중을 차지한다[2].

소프트웨어 제품라인 개발에서 제품군을 생성하기 위해 사용되는 소스코드를 제품라인 코드 베이스라고 부른다. 소프트웨어 제품라인을 위한 회귀시험은 제품라인 코드 베이스가 변경되었을 때 제품군을 재시험하는 시험활동을 말한다. 제품라인 회귀시험을 수행하기 위해, 제품군이 변경되기 전에 통과(pass)한 기존 시험 항목들을 모두 재실행할 수 있다. 그러나 이 방법은 제품군이 변경될 때마다 변경과 무관한 시험 항목들이 불필요하게 실행되기 때문에 매우 비효율적이다. 따라서 제품라인 코드 베이스의 변경에 영향을 받은 시험 항목들을 식별하여 선택하고 이들 만을 재실행함으로써 불필요한 시험 비용을 줄일 필요가 있다. 이런 목적으로 소프트웨어의 변경된 부분을 재시험하기 위해 기존 시험 항목들 중 어떤 시험 항목들을 선택해야 하는가 하는 문제를 회귀시험 항목 선택 문제(Regression test selection)라고 한다[1].

회귀시험 항목 선택 문제를 다룬 방법으로는 지금까지 아키텍처 기반의 제품라인 회귀시험 항목 선택 방법들[3][4]이 제안되었다. 그러나 이 방법들은 아키텍처가 철저히 관리된 제품라인에서만 적용 가능 하거나[4] 또는 시험 전문가의 개입을 요구 한다[3]. 따라서 과거 연구

† 본 연구는 한국연구재단의 지원(NRF-2017M3C4A7066210)으로 수행하였음

‡ 비회원 : 삼성전자 DIT센터
psjung416@gmail.com
§ 종신회원 : 한국과학기술원 전산학부 교수
sungwon.kang@kaist.ac.kr

논문접수 : 2020년 09월 13일
심사완료 : 2020년 11월 02일

```
1  public class Calculator{
2      public double sum(double n1, double n2){
3          return n1 + n2;
4      }
5      public double sub(double n1, double n2){
6          return n1 - n2;
7      }
8      public double mult(double n1, double n2){
9          return n1 * n2;
10     }
11     public double div(double n1, double n2){
12         return n1 / n2;
13     }
14 }
1  public class TestCalculator{
2      @Test
3      public void t1(){
4          assertEquals(
5            10.0, mult(div(sum(3.0,7.0),
6                        sub(4.3, 2.3)), 2.0));
7      }
8  }
```

P1

```
1  public class Calculator{
2      public double sum(double n1, double n2){
3          return n1 + n2;
4      }
5      public double sub(double n1, double n2){
6          return n1 - n2;
7      }
8      public double div(double n1, double n2){
9          return n1 / n2;
10     }
11     public double mod(double n1, double n2){
12         return n1 % n2;
13     }
14 }
1  public class TestCalculator{
2      @Test
3      public void t2(){
4          assertEquals(
5            1.0, mod(div(sum(3.0,7.0),
6                        sub(4.3, 2.3)), 2.0));
7      }
8  }
```

P2

```
1  public class Calculator{
2      public double sum(double n1, double n2){
3          return n1 + n2;
4      }
5      public double sub(double n1, double n2){
6          return n1 - n2;
7      }
8      public double mult(double n1, double n2){
9          return n1 * n2;
10     }
11     public double mod(double n1, double n2){
12         return n1 % n2;
13     }
14 }
1  public class TestCalculator{
2      @Test
3      public void t3(){
4          assertEquals(
5            0.0, mult(mod(sum(3.0,7.0),
6                        sub(4.3, 2.3)), 2.0));
7      }
8  }
```

P3

그림 2. 계산기 제품군의 소스코드

중에는 자동화된 소스코드 기반의 회귀시험 항목 선택 방법은 없었고, 이러한 방법이 최초로 논문 [5]에서, 제품라인 코드 베이스에 변경이 발생했을 때 회귀시험을 효율적으로 수행하는 자동화된 방법 SRTS(Space based Regression Test Selection for software product lines)으로 발표되었다.

본 논문은 SRTS방법을 소개한다. 이 방법은, 먼저 제품라인 코드 베이스와 시험 항목을 공통성과 가변성을 기반으로 나누어 파티션 테이블을 만들고 제품라인 코드 베이스의 변경에 영향을 받은 시험 항목을 테이블로부터 식별하여 선택한 후, 선택된 시험 항목만을 재실행함으로써 불필요한 시험을 줄인다.

본 논문의 구성은 다음과 같다. 제 2 절에서는 제품라인 회귀시험의 개요를 소개한다. 제 3 절에서는 SRTS방법을 소개한다. 끝으로, 제 4절에서는 SRTS 방법의 의의를 설명하고 향후 연구 방향을 제시한다.

## 2. 제품라인 회귀시험

제품라인의 제품군은 가변성 모델과 바인딩 정보를 기반으로 정의되고 생성된다. 가변성 모델은 제품군의 공통성과 가변성을 표현하는 모델이다. 제품라인 내의 공통적인 피처를 필수적(Mandatory) 피처로 표현하고 제품 간의 구별되는 차이점을 선택적(Optional) 피처, 택일적(Alternative) 피처 등으로 표현한다. 그림 1(a)는 계산기 제품라인의 가변성 모델을 보여준다. 덧셈 연산과 뺄셈 연산을 포함한 공통 소스코드 c1은 모든 제품의 코드에 공통적으로 포함된다. 그러나 특수 연산(Special op)을 위한 제품의 코드는 c2 와 c3를 포함하거나 c3 와 c4를 포함하거나 c2 와 c4 를 포함하고, 제품 시험을 위해 적용되는 시험 항목은 t1, t2, t3 중 하나가 된다. 바인딩은 제품라인의 공통 산출물로부터 각 제품 별 산출물 인스턴스를 생성하기 위해 반드시 필요한 정보로서, 이로부터 제품라인 코드 베이스 및 시험 항목 중 어떤 부분이 각 제품에 포함되는지 알수있다. 그림 1(b)는 계산기 제품군을 생성하기 위한 바인딩 정보를 보여준다. 그림 1(b)에서 제품 P1을 생성하기 위한 바인딩 정보는BP1인데, BP1은 특수 연산을 위한 구현으로 c2와 c3를 바인드 해주고 시험 항목으로는 t1을 바인드 해준다.

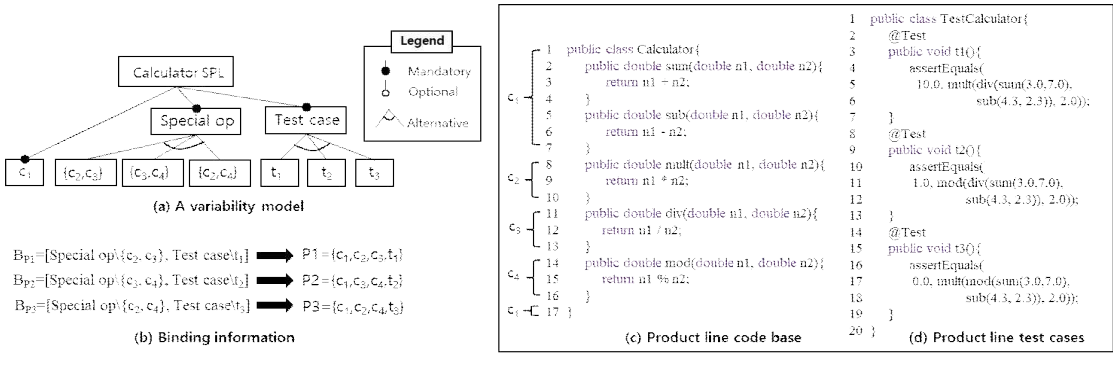



그림 1. 가변성 모델, 바인딩 정보, 제품라인 코드 베이스, 제품라인 시험 항목

그림 1(c)는 제품라인 코드 베이스, 그림 1(d)는 제품라인 시험 항목을 보여준다. 이들은 각각 제품군을 생성하기 위해 필요한 모든 소스코드를 포함하고, 제품군을 시험하는 모든 시험 항목들을 포함한다. 제품군의 소스코드와 시험 항목은 제품라인 코드 베이스와 제품라인 시험 항목에 바인딩 정보를 적용하여 생성된다. 예를 들어, 그림 1(b)의 P1을 위한 바인딩 정보BP1를 제품라인 코드 베이스와 제품라인 시험 항목에 적용하면, 제품라인 코드 베이스의 c1, c2, c3가 조합되어 P1이 생성되고 시험 항목 t1이 P1의 시험을 위해 선택된다. 그림 2는 제품라인으로부터 생성된 제품군의 소스코드와 시험 항목을 보여준다. 제품군이 생성되면 각 제품은 해당 제품을 위해 선택된 모든 시험 항목을 실행함으로써 시험된다.

제품라인에 새로운 피처가 추가되거나 기존 피처가 수정되면 제품라인 코드 베이스가 변경될 수 있다. 변경된 제품라인 코드 베이스를 재시험하는 가장 간단한 방법은 기존 시험 항목들을 모두 재실행하는 전수 시험(Retest-all)을 하는 것이다[4]. 소프트웨어 제품라인의 회귀 시험을 위한 전수 시험 방법은 새로운 결함을 탐지하는데 효과적이지만 변경에 무관한 많은 시험 항목들을 실행하게 되므로 매우 비효율적이다.

전수 시험의 대안으로, 단일 제품을 위한 기존의 회귀 시험 방법을 제품군의 각 제품에 적용할 수 있는데, 이 회귀 시험을 스토브파입 시험(Stove-pipe Test)이라고 부르기로 한다. 이 방법은 변경에 영향을 받지 않는 시험 항목을 각 제품 별로 식별하여 배제함으로써 전수 시험에 비해 제품군의 회귀시험 비용을 많이 줄일 수 있다. 그러나 코드 기반 회귀시험 방법들은 소스코드와 시험 항목을 분석하기 위해 많은 비용을 요구하고, 단일 제품을 위한 회귀시험 절차를 제품 수만큼 반복 적용하게 되기 때문에 제품 수가 많은 제품군에서는 실용적이지 않다.

## 3. 효율적인 소프트웨어 제품라인 회귀시험 방법

하나의 제품 라인 시험 항목은 동일한 소스코드를 갖는 제품들을 시험하기 위해 재사용될 수 있다. 하지만 서로 다른 소스코드를 시험하기 위해 다수의 제품에 재사용되는 경우, 제품마다 다른 시험 산출물(예를 들어, 시험 커버리지, 시험 결과 등)을 생성할 수 있기 때문에 재사용의 효과가 크게 줄어들 수 있다. 또한, 시험 항목이 멀티쓰레딩, 비동기 함수, 랜덤 함수 등과 같은 비결정적인 기능을 시험하는 경우, 그 시험 항목은 동일한 제품에서 실행될 때마다 다른 결과를 낼 수 있다. 따라서 본 연구는 SRTS를 소개하기 이전에 이 두 가지 측면들에 대해 다음의 가정을 한다.

**제품라인 시험 항목의 결정적 실행 가정.** 제품라인 시험 항목은 1) 같은 제품에서 여러 번 실행될 때 항상 같은 코드 부분을 시험하고 같은 시험 결과를 산출한다. 또한 2) 그 시험 항목을 사용하는 제품들에서 각각 실행될 때 항상 같은 코드 부분을 시험하고 같은 시험 결과를 산출한다.

위 가정의 1)은 개별 제품에 대한 시험 항목의 결정적 실행에 대한 가정이고, 2)는 제품군에 대한 시험 항목의 결정적 실행에 대한 가정이다. 가정의 2)와 관련하여, 이론적으로는 하나의 시험 항목이 서로 다른 두 제품에서 다른 코드 부분을 시험하더라도 재사용 가능하다. 그러나 이 경우, 시험 항목의 실행 결과가 제품마다 달라질 수 있고, 제품라인 코드 베이스가 변경되었을 때 시험 항목의 관리가 매우 복잡해진다. 따라서 시험 항목은 적용될 때 동일한 코드 부분을 시험하는 제품들에서만 재사용되어야 한다고 가정한다.

제품라인의 회귀시험을 효율적으로 수행하기 위해, SRTS는 공통성과 가변성을 기반으로 제품라인 소스코드와 시험 항목들을 나눈다. 바인딩 정보를 이용하여 각 소스 코드 조각이 포함된 제품 집합을 식별하면 제품라인 코드 베이스와 시험 항목들을 자동으로 여러 개의 코드 영역과 시험 스위트로 나뉜다. 그림 3과 4는 7개의 코드 영역으로 나눠진 제품라인 코드 베이스와 시험 항목을 보여준다. 예를 들어, 그림 1(b)의 바인딩 정보에 따라, c1은 모든 제품에 포함되므로 코드 영역 G에 포함되고, c2는 P1과 P3에 포함되므로 코드 영역 E에 포함되고, c3

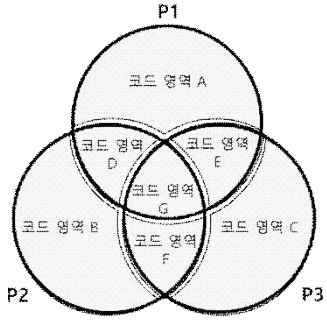



그림 3. 공통성과 가변성을 기반으로 나누어진 제품라인 코드 베이스

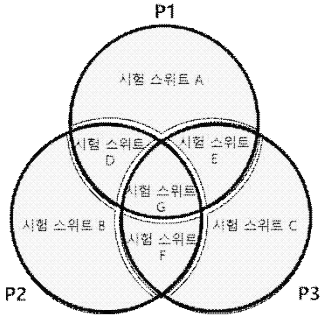



그림 4. 공통성과 가변성을 기반으로 나누어진 제품라인 시험 항목

는 P1과 P2에 포함되므로 코드 영역 D에 포함된다. 마찬가지로, t1은 P1에만 포함되므로 시험 스위트 A에 포함되고, t2는 P2에만 포함되므로 시험 스위트 B에 포함된다. 우리는 모든 제품에 공통으로 포함되는 코드 영역의 소스코드(예를 들어, 그림 3의 코드 영역 G)를 공통 소스코드라고 부르고 그 외의 소스코드(예를 들어, 그림 3의 코드 영역 A, B, C, D, E, F)를 가변 소스코드라고 부른다.

### 3.1. 공통 소스코드의 변경을 위한 회귀시험

Ekstazi[6]는 단일 제품을 위한 최신의 자동화된 코드 기반 회귀시험 방법이다. 이 방법은 각 시험 항목의 커버리지 정보를 수집한 후, 이를 활용하여 제품 변경 시 변경된 소스코드를 시험하는 시험 항목을 식별하여 재실행한다. 시험 항목의 커버리지 정보를 수집하기 위해, 프로그램에 바이트 코드를 삽입(BCI: Byte-Code Instrumentation)하여 시험 항목이 실행될 때 커버리지 정보가 수집되도록 한다. 그러나 BCI는 많은 비용을 요구하기 때문에 제품군의 회귀시험을 위해 각 제품에 BCI를 반복적으로 수행하는 것은 실용적이지 않다. 따라서 한 제품에서 수집한 커버리지 정보를 다른 제품에 재사용하여 BCI의 반복 수행을 줄일 필요가 있다.

그림 5와 같이, t1, t2, t3가 회귀시험을 위해 선택되고 각 시험 항목이 시험하는 제품이 화살표의 방향과 같을 때, P1 과 P2 에 BCI 를 수행하면 모든 시험 항목의 커버리지 정보와 시험 결과를 얻을 수 있다. P3을 회귀시험하기 위해서 t3가 필요하지만, 제품라인 시험 항목의 결정적 실행을 가정하면 t3는 P1으로부터 얻은 커버리지 정보와 시험 결과가 P3로부터 수집되는 것과 같을 것이므로 재사용될 수 있다. 따라서 P3 에 BCI 를 수행할 필요가 없고 t3는 P3에서 실행될 필요가 없다.

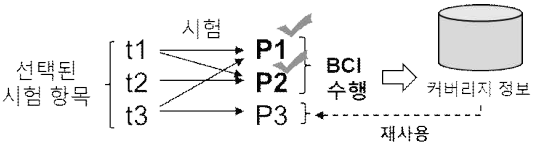



그림 5. 제품군의 회귀시험을 위한 바이트 코드 삽입

### 3.2. 가변 소스코드의 변경을 위한 회귀시험

하나의 시험 항목은 두 개 이상의 제품에서 동시에 실행될 수 없다. 다시 말해서, 하나의 시험 항목은 단 하나의 제품을 구성하는 코드 영역들에서만 실행될 수 있다. 이 사실을 이용하면 제품군의 소스코드와 시험 항목의 분석 없이 회귀시험 항목수를 효과적으로 줄일 수 있다. 예를 들어, 만약 코드 영역 A가 변경되어 회귀시험을 수행해야 하는 경우, 코드 영역 A는 P2와 P3에 포함되지 않는 코드 영역이므로 P2와 P3에서 실행 가능한 시험 항목들은 코드 영역 A를 절대로 시험하지 않을 것이다. 따라서 시험 스위트 B, C, D, E, F, G는 변경에 무관하므로 제거할 수 있다. 마찬가지로, 코드 영역 F가 변경된 경우, 코드 영역 F는 P1에 포함되지 않으므로 P1에서 실행가능한 시험 항목들은 코드 영역 F를 절대로 시험하지 않을 것이다. 따라서 시험 스위트 A, D, E, G를 제거할 수 있다. 표 1은 각 코드 영역이 변경될 때, 회귀시험을 위해 제거할 수 있는 시험 스위트들을 보여준다.

표 1. 그림 3의 제품군에 대한 회귀시험 선택

| 변경된 코드 영역 | 제거될 시험 스위트 |
|---|---|
| 코드 영역 A | 시험 스위트 B, C, D, E, F, G |
| 코드 영역 B | 시험 스위트 A, C, D, E, F, G |
| 코드 영역 C | 시험 스위트 A, B, D, E, F, G |
| 코드 영역 D | 시험 스위트 C, E, F, G |
| 코드 영역 E | 시험 스위트 B, D, F, G |
| 코드 영역 F | 시험 스위트 A, D, E, G |
| 코드 영역 G | - |

이 접근 방법은 가변 소스코드의 변경을 위한 회귀시험에는 효과적이지만 공통 소스코드가 변경되었을 경우에는 효과적이지 않다. 예를 들어, 코드 영역 G가 변경되었을 경우에는 이 방법을 사용해서 제거할 수 있는 시험 스위트가 없다. 따라서 3.1절에서 설명한 회귀시험 항목 선택 방법을 적용하는 것이 좋다.

### 3.3. 방법의 평가

이 절의 SRTS의 평가는 논문 [5]에 나오는 내용을 요약한 것이다. SRTS의 검증을 위해, SPL2go[7] 오픈 소스 저장소로부터 Java 언어로 구현된 여섯 개의 제품라인 프로젝트를 선정하였다. 표 2는 대상 제품라인의 이름과 각 제품라인으로부터 생성한 제품군의 코드 규모(SLOC), 제품군의 크기(|PSPL|), 공통 소스코드의 비율(Com.), 시험 항목의 수(|TSPL|), 시험 항목의 총 초 단위의 실행 시간(ET)을 보여준다.

표 2. 대상 제품라인과 제품군

| 대상 제품라인 | SLOC | $\|P_{SPL}\|$ | Com. | $\|T_{SPL}\|$ | ET |
|---|---|---|---|---|---|
| MobileMedia | 3,196 | 4 | 44.2% | 1,139 | 6.9 |
| TankWar | 4,845 | 5 | 62.9% | 782 | 2.6 |
| Prevayler | 5,109 | 3 | 72.5% | 1,306 | 9.9 |
| | | 5 | 72.5% | 2,543 | 23.2 |
| MobileRSS Reader | 16,664 | 3 | 93.1% | 3,247 | 11.1 |
| | | 5 | 93.1% | 5,383 | 22.2 |
| Lampiro | 30,158 | 4 | 98.8% | 6,236 | 128.3 |
| | | 6 | 98.6% | 9,346 | 188.0 |
| BerkeleyDB | 44,994 | 3 | 69.3% | 10,297 | 23.0 |
| | | 5 | 69.3% | 18,407 | 46.8 |
| | | 7 | 69.3% | 26,290 | 55.7 |

대상 제품라인의 시험 항목은 EvoSuite 도구[8]를 활용하여 생성하였고, 제품군의 변경된 버전은 PIT 도구[9]를 활용하여 제품라인 별로 50개를 생성하였다.

SRTS의 효용성을 검증하기 위해, 단일 제품을 위한 회귀시험 방법을 제품군에 적용하는 전통적인 제품군 회귀시험 방법과 비교하였다. 단일 제품을 위한 회귀시험 방법은 Ekstazi[6]를 채택하고, 이를 제품군의 각 제품에 적용하는 방법을 Ekstazi_SPL이라고 부른다. Ekstazi는 변경에 영향을 받는 시험 항목을 모두 선택하는 방법이고, Ekstazi_SPL은 제품군의 각 제품에 Ekstazi를 반복 적용하는 것이므로 Ekstazi_SPL 또한 변경에 영향을 받는 시험 항목을 모두 선택한다. SRTS와 Ekstazi_SPL의 비교 항목은 선택된 결함 노출 시험 항목(fault-revealing test case)의 수, 선택된 시험 항목의 총 개수, 총 회귀시험 수행 시간(회귀 시험 항목을 선택하고 선택된 시험 항목을 실행하는 시간)이고 50개의 변경된 버전을 대상으로 실험한 결과의 평균 값을 비교하였다.

실험 결과, SRTS와 Ekstazi_SPL 방법은 모든 결함 노출 시험 항목을 선택하였다. 이 결과가 나타난 이유는 변경된 코드를 시험하는 시험 항목을 모두 선택하는 방법은 항상 모든 결함 노출 시험 항목을 선택하게 되고 [10], 두 방법은 모두 제품라인 시험 항목의 결정적 실행 가정 하에 위 조건을 만족하는 방법들이기 때문이다.

표 3은 대상 제품군에 대해, 선택된 시험 항목의 총 개수, 회귀시험 총 시간에 관하여 SRTS와 Ekstazi_SPL 방법을 비교한 결과이다. 표 3에서 괄호 안의 숫자는 제품군을 구성하는 제품의 수이다. SRTS는 모든 대상 제품군에 대해 Ekstazi_SPL보다 총 회귀시험 시간을 14.8% ~ 49.1% 줄였다. 특히, Prevayler(3), Prevayler(5), MobileRSSReader(3), Lampiro(4) 제품군에서 적은 수로 나눠진 코드 영역으로 인해 SRTS가 Ekstazi_SPL보다 적은 수의 시험 항목을 줄였음에도 불구하고 더 많은 회귀시험 시간을 줄였다. 이러한 결과는 SRTS가 회귀시험 절차의 불필요한 반복을 줄이고 소스코드와 시험 항목의 불필요한 분석을 효과적으로 줄인다는 것을 의미한다.

## 4. 결론

본 논문은 소스코드를 기반으로 제품라인 회귀시험을 효율적으로 수행하는 자동화된 방법인 SRTS를 소개하였다. SRTS는 공통 소스코드의 변경과 가변 소스코드의 변경을 별도로 다룬다. 공통 소스코드가 변경되었을 때 한 제품의 시험 산출물을 재사용하여 불필요한 회귀시험 절차의 반복을 줄이고, 가변 소스코드가 변경되었을 때 제품군의 제품라인 코드베이스와 시험 항목을 공통성과 가변성을 기반으로 분할하여 소스코드와 시험 항목의 분석 없이 변경에 무관한 시험 항목들을 제거한다.

실험 결과에 나타나듯이, SRTS를 적용하면 시험 항목을 선택하고 선택된 시험 항목을 실행하는 총 회귀 시험 시간을 Ekstazi_SPL과 비교하여 4.8%~49.1%만큼 줄일 수 있다. 이 효과는 제품군을 구성하는 제품 수가 많을수록, 제품 간의 공통부분이 많을수록 커지는 경향을 보인다. 따라서 SRTS는 다수의 유사한 제품들을 효율적으로 개발하기 위해 개발 산출물을 최대한 많이 재사용한 제품군에서 큰 효과를 보인다.

본 논문에서 소개한 SRTS에 관련된 후속 연구로는 SRTS의 효용성을 대규모의 제품라인을 대상으로 검증하는 연구와 SRTS를 확장하여 회귀시험을 위해 선택된 시험 항목들을 실행할 때 발생하는 불필요한 비용을 줄이는 연구가 필요하다.

표 3. 실험 결과

| 대상 제품라인 | 코드 영역 수 | 선택된 시험 항목의 수 | | | 총 회귀시험 수행 시간 | | |
|---|---|---|---|---|---|---|---|
| | | SRTS | Ekstazi_SPL | Savings | SRTS | Ekstazi_SPL | Savings |
| MobileMedia(4) | 11 | 10.7% | 12.7% | 15.6% | 28.0% | 39.0% | 28.1% |
| TankWar(5) | 17 | 28.1% | 34.5% | 18.7% | 80.5% | 158.0% | 49.1% |
| Prevayler(3) | 4 | 29.2% | 20.0% | -46.0% | 48.6% | 59.1% | 17.1% |
| Prevayler(5) | 6 | 31.4% | 23.4% | -34.3% | 49.5% | 58.1% | 14.8% |
| MobileRSSReader(3) | 5 | 23.4% | 23.4% | -0.2% | 58.4% | 87.6% | 33.3% |
| MobileRSSReader(5) | 16 | 22.9% | 23.4% | 1.8% | 48.0% | 84.8% | 43.4% |
| Lampiro(4) | 10 | 16.3% | 16.2% | -0.6% | 26.2% | 36.4% | 28.0% |
| Lampiro(6) | 16 | 16.1% | 16.3% | 1.2% | 24.2% | 37.7% | 35.9% |
| BerkeleyDB(3) | 4 | 24.4% | 25.4% | 3.9% | 61.1% | 71.7% | 14.8% |
| BerkeleyDB(5) | 16 | 24.0% | 32.4% | 26.0% | 55.5% | 75.0% | 26.0% |
| BerkeleyDB(7) | 31 | 20.9% | 32.0% | 34.6% | 50.8% | 72.1% | 29.6% |

## 참 고 문 헌


[1] Yoo, S., Harman, M., 2012. Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability. 22 (2), 67 - 120.

[2] Chittimalli, P. K., & Harrold, M. J., 2009. Recomputing coverage information to assist regression testing. IEEE Transactions on Software Engineering, 35(4), 452-469.

[3] Neto, P.A.d.M.S., do Carmo Machado, I., Cavalcanti, Y.C., de Almeida, E.S., Garcia, V.C., de Lemos Meira, S.R., 2010. A regression testing approach for software product lines architectures. In: Proceedings of the Fourth Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), pp. 41 - 50.

[4] Lity, S., Nieke, M., Thüm, T., & Schaefer, I., 2019. Retest test selection for product-line regression testing of variants and versions of variants. Journal of Systems and Software, 147, 46-63.

[5] Jung, P., Kang, S., & Lee, J., 2019. Automated code-based test selection for software product line regression testing. Journal of Systems and Software, 158, 110419.

[6] Gligoric, M., Eloussi, L., Marinov, D., 2015. Practical regression test selection with dynamic file dependencies. In: Proceedings of the International Symposium on Software Testing and Analysis, pp. 211 - 222.

[7] SPL2go. Available online: http://spl2go.cs.ovgu.de/ (accessed on September 2020).

[8] Fraser, G., Arcuri, A., 2011. EvoSuite: automatic test suite generation for object-oriented software. Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, pp. 416-419.

[9] PIT. Available online: http://pitest.org/ (accessed on September 2020).

[10] Rothermel, G., Harrold, M.J., 1996. Analyzing regression test selection techniques. IEEE Transactions on Software Engineering. 22 (8), 529 - 551.



정 필 수

2012년 충남대학교 컴퓨터공학과 졸업 (학사)

2014년 한국과학기술원 전산학과 졸업 (석사)

2020년 한국과학기술원 전산학과 졸업 (박사)

관심분야는 소프트웨어 아키텍처, 소프트웨어 제품라인 공학, 소프트웨어 재사용



강 성 원

1982년 서울대학교 사회과학대학 졸업

1989년 University of Iowa 전산학 석사

1992년 University of Iowa 전산학 박사

1993 ~ 2001년 한국통신(KT) 선임연구원

2001 ~ 2009년 한국정보통신대학교 전산학과 교수

2009 ~ 현재 한국과학기술원 전산학부 교수

# 델파이기법을 이용한 감리점검항목 도출 방안†

## - 하이퍼레저 패브릭 기반으로

## (An Audit Method on Information System Audit using Delphi Method - Based on Hyperledger Fabric)

이 영 주 ‡ 박 수 용 §

(Youngjoo Lee) (Sooyong Park)

**요 약**

국내 공공기관에서 사업비가 5억원 이상인정보시스템 구축사업은 정보시스템 감리대상이고 정보시스템 감리원은 감리기준에 따라 감리를 실시하여야 한다. 최근 금융산업을 넘어 제조, 헬스케어, 유통, 공공 부문 등 다양한 분야에 블록체인 기술이 적용되면서 블록체인 기술을 적용한 시스템 개발이 늘고 있다. 공공기관과 기업에서는 블록체인 기술을 적용하기 위해 프라이빗 블록체인인 하이퍼레저 패브릭을 사용이 증가 추세에 있다. 그러나, 새롭게 부상하고 있는 블록체인기반 시스템을 기존의 감리점검항목으로 감리를 수행하기에는 충분하지 않아 각 감리원들이 검증되지 않은 점검항목을 개별적으로 만들어 사용하고 있다. 따라서 블록체인을 적용한 시스템을 감리할때 검증된 감리점검 항목이 필요하다. 본 연구에서는 델파이 프로세스를 커스터마이징하여 블록체인 기술을 활용한 시스템 개발사업에 적합한 점검항목을 도출하고, 정보시스템 감리원의 설문조사를 통해 도출된 점검항목의 적정성을 검증받았다. 이번 연구는 정보시스템 감리원의 감리 수행에 직접적인 도움이 될 것이며, 블록체인 기반 시스템을 통해 서비스를 제공하는 사업자와 주관사도 시스템의 품질 향상에 기여할 것으로 기대된다.

**키워드** : 정보시스템감리, 델파이기법, 소프트웨어공학, 하이퍼레저 패브릭, 블록체인

**Abstract** A project to establish an information system with a project cost of more than 500 million won at a local public firm is subject to the Information System Audit (hereinafter referred to as IS Audit), and the IS Auditor shall conduct audit according to the audit criteria. Recently, as blockchain technology has been applied to various fields such as manufacturing, healthcare, distribution, and public sectors beyond the financial industry, the development of systems that apply blockchain technology is increasing. The use of Hyperledger Fabric, a private blockchain, is on the rise to utilize blockchain technology in public firms and private firms. However, the newly emerging blockchain-based system is not sufficient to carry out auditing with existing audit check items, so it has no choice but to make and use audit items individually. Therefore, a need for verified audit items for systems that base on blockchain technology has emerged. In this study, we customized the Delphi process to derive audit items suitable for the system development project using the blockchain technology, and verified the completeness and accuracy of the audit items derived through a survey by the IS Auditor. This research will be of direct help to the IS Auditor, and it is expected that operators and organizers who provide services through the blockchain-based system will also contribute to improving the quality of the system.

**Key words** : Information System Audit, Software Engineering, Delphi Method, Blockchain, Fabric

## 1. 서 론

정보시스템감리는 미국에서 민간중심으로 감리가 발전하기 시작하였고, 일본에서는 정부 주도로 감리가 발전하기 시작했다. 일본의 경우 1985년 통산성에서 시스템감사 기준을 제정 공표한 이래, 1999년 통상산업성에서 '시스템 감사인 인정제도'를 도입하여 시행하고 있다. 우리나라에서 정보시스템감리는 1986년 한국전산원(현 한국정보화진흥원)에서 처음 실시 한 이후, 1998년 민간감리법인 설

† 이 논문은 정부(정보통신기술진흥센터)의 재원으로 대학ICT연구센터육성지원사업의 지원을 받아 수행된 연구임 (No. IITP-2020-2017-0-01628).

‡ 학생회원 : 서강대학교 컴퓨터공학과
yjoo99@hanmail.net
§ 종신회원 : 서강대학교 컴퓨터공학과 교수
sypark@sogang.ac.kr

논문접수 : 2020년 10월 06일
심사완료 : 2020년 11월 3일

립이 시작했다. 그리고, 1999년 “정보화촉진기본법” 제15조의 2에 정보시스템 감리의 근거 조항이 신설되었고(현재 전자정부법 57조,시행령 제71조), 이에 근거하여 “정보시스템 감리기준”(1999년 제정)이 고시되었다. 정보시스템 감리기준에는 감리의 계약,감리계획수립, 착수회의, 현장감리, 감리보고서작성, 감리결과 조치내역확인 등 감리의 절차 및 방법을 규정하고 있다. 정보시스템구축사업으로 사업비가 5억원 이상인 경우 감리대상이다.

“정보시스템 감리란 감리발주자 및 피감리인의 이해관계자로부터 독립된 자가 정보시스템의 효율성을 향상시키고 안전성을 확보하기 위하여 제3자의 관점에서 정보시스템의 구축 및 운영 등에 관한 사항을 종합적으로 점검하고 문제점을 개선하도록 하는 것을 말한다.[1]

공공기관의 시스템개발사업 수행할때 시스템이 기술적으로 지켜야할 의무사항이 다수 있다. 예들들어, 전자정부법 50조 시행령59조를 근거로 공공기관의 데이터베스표준화 지침을 준수해야 한다. 국가정보화기본법 제 32조(장애인·고령자 등의 정보접근 및 이용보장)와 장애인차별금지법 제 21조(정보통신·의사소통에서의 정당한 편의제공의무)와 시행령 제 14조(정보통신·의사소통에서의 정당한 편의제공의 단계적 범위 및 편의의 내용)에 근거하여 웹접근성을 준수해야 한다. 또한, 전자정부법 제50조(전자정부서비스 호환성 준수지침)에 따라 행정기관등의 장은 웹 사이트 호환성 확보, 웹 표준 준수 및 비표준 기술 제거와 같은 전자정부 서비스 호환성을 1년에 한 번 이상 진단해야 한다.

이밖에도 전자정부법 45조 제3항 행정기관 및 공공기관 정보시스템 구축·운영 지침 (제6장 51조,52조,53조)에 따라 설계단계에 20개 항목과 구현단계에 47개의 보안약점이 없도록 소프트웨어 보안약점이 없도록 소프트웨어를 개발 또는 변경해야 한다. 개인정보보호법 제23조제2창, 제24조제3항 및 제29조에 따라 개인정보는 암호화, 파기 접속기록보관에 따른 기준을 준수하고, 정보통신망법의 개인정보의 기술적·관리적 보호조치 기준에 따라 개인정보를 보호하여 시스템개발을 해야한다.

위와같이 공공기관의 시스템개발사업을 수행할 경우 지켜야할 기술적 의무사항이 다수 있다. 정보시스템 감리원은 이와같이 법에서 규정한 사항이 시스템에 적합하게 이행되었는지 점검해야한다. 그런데 블록체인 기술을 적용한 시스템구축 사업에서는 위와같은 법규정을 적용하려면 새로운 점검항목이 필요하다.

그러나, 현재의 감리 점검프레임워크 및 감리수행점검가이드는 2013년 이후로 개정되지 않았고 현행 정보시스템 감리점검해설서 V3.0에서는 블록체인과 관련된 점검항목이 부재하다. 한국정보화진흥원은 기존 감리체계의 문제점을 개선하고자 2016년 정보시스템 규모.유형별 감리 참조 모델(안)을 제시하였고 기본 점검항목 대비 검토항목이 대부분 1:1로 구성되어 세분화가 미흡하여 재정비가 필요하다는 의견이 포함되었다.[17] 하지만, 2020년 현재까지 점검항목은 아직 구체화되지 않은 상태이다.

본 논문에서는 이와 같은 문제점을 해결하여 블록체인 기술을 적용한 시스템 구축 사업에 대한 감리 점검항목을 도출하여 객관적인 감리활동에 도움이되고자 한다. 또한 블록체인과 같은 신기술에 대한 감리점검항목을 도출하기 위한 방법으로, 커스터마이징된 델파이 프로세스를 적용한 결과가 적정하다는 것을 보여줌으로써 검증된 점검항목 도출방안을 제시하고자 한다.

연구범위는, 일반적으로 시스템개발사업에서 감리영역은 사업관리, 응용시스템, 데이터베이스, 시스템아키텍처 4개 분야이지만, 개발방법론과 기술에 영향을 받는 응용시스템과 데이터베이스 영역으로 제한하였다. 그리고, 사업영역은 하이퍼레저 패브릭기반의 시스템개발사업을 대상으로 제한다.

본 논문의 구성은 1장 서론에서는 본 연구의 필요성에 대해서 기술하고, 2장에서는 정보시스템 감리 관련연구와 배경지식인 하피퍼레저 패브릭을 기술하고, 3장에서는 기존 감리점검항목의 문제점과 델파이기법을 이용한 블록체인기반의 시스템개발사업에서의 감리점검항목을 제시하였고, 4장에서는 도출된 점검항목에 대한 적정성을 검증하였다. 마지막으로 제 5장에서는 본 연구의 요약 및 결론을 기술하였다.

## 2. 배경 지식 및 관련 연구

### 2.1 배경 지식

본 연구에서는 이해를 돕기 위해서 생소할 수 있는 정보시스템감리 소개와 절차와 정보시스템 감리 점검 프레임워크, 감리 점검항목에 대해 다룬다. 또한, 하이퍼레저 패브릭에 대한 개념을 간략하게 설명한다.

#### 2.1.1 정보시스템감리 개요

정보시스템감리는 요구정의단계감리, 설계단계감리, 종료단계감리 3단계로 이루어지며 20억이하이거나 6개월이하 사업인 경우 요구정의단계감리는 생략가능하다.

단계별감리는 예비조사, 현장감리, 시정조치확인의 순서로 이루어지며, 각 절차가 종료된 후 결과물로 감리계획서, 감리수행결과보고서, 시정조치확인보고서를 각각 작성하여 제출해야 한다. 현장감리는 감리시작, 착수회의,감리수행, 보고서 작성 및 검토, 종료회의, 보고서 확정및 통보의 6가지 절차로 구성된다.[3]

감리수행 방법은 단계별 감리를 실시하는 단계는 “정보시스템 감리 점검프레임워크 V3.0”의 “사업유형/감리시점” 등을 참고하여 정하는 것을 원칙으로 한다.

감리수행 결과물은 감리수행결과보고서와 시정조치확인보고서 2종류가 있다. 본 감리에서 감리수행결과보고서를 작성하고, 본 감리에서 개선 방향으로 나온 내용에 대해 시정조치를 하고 그 결과를 감리시정조치확인보고서에 작성한다. 아래의 그림은 감리 점검 프레임워크이다.

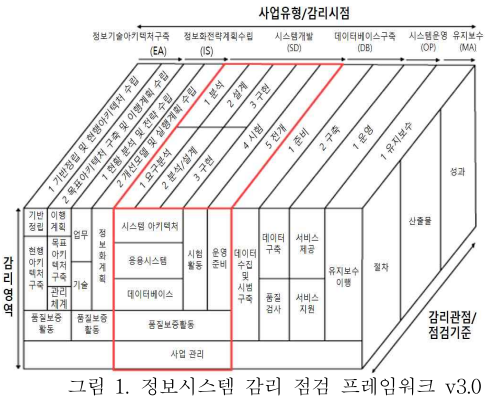



그림 1. 정보시스템 감리 점검 프레임워크 v3.0

### 2.1.2 블록체인 기술[16]

블록체인은 탈중앙화된 분산 시스템이고 기존의 중앙집중식 거래 시스템에 비해 다음과 같은 몇가지 기술적 특징이 있다.

- 암호 : 블록체인에 기록된 거래는 공개키과 개인키 쌍을 이용해 암호화된다.
- 실시간 : 거래가 거의 발생하자마자 블록체인에 저장되기 때문에 거의 실시간 거래 기록과 계정 대조조정(reconciliation of accounts)
- 스마트 계약 호스팅 : 블록체인은 프로그래밍 코드를 내장해 스마트 컨트랙트를 포함한다. 이들 프로그램은 특정 계약조건이 발동될 때 거래를 실행하고 해당 원장 항목을 작성할 수 있다.

블록체인은 비트코인과 같은 누구나 참여할 수 있는 허가형(Permissioned)과 특정 참가자만 접근할수 있는 비허가형(Permisionless blockchain) 두 개의 타입이있다. 비허가형 블록체인의 경우 각 참여자가 동일한 블록체인을 유지관리하고 모든 복사본을 지속적으로 동기화하여 데이터의 투명성, 정확성 및 최신 상태를 보장함으로써 합의가 달성된다. 허가형 블록체인은 일부 참가자만 사본의 일부를 가질 수 있다. 접근제어구성에 따라 특정 참가자와 특정정보를 제한할 수있다.

### 2.1.3 하이퍼레저 패브릭(Hyperledger Fabric):

하이퍼레저 패브릭은 허가형 프라이빗 블록체인(Permissioned Private Blockchain)의 형태를 가진다. 누구나 자유롭게 참여가 가능한 기존의 퍼블릭 블록체인과 달리, 하이퍼페저 패브릭에서는 인증 관리시스템(MSP)에 의해 허가된 사용자만이 블록체인 네트워크에 참여할 수 있다. 따라서 패브릭 네트워크에 참여한 노드들은 이미 시스템에 의해 허가된, 신뢰를 가진 노드로 볼 수 있고 퍼블릭 블록체인에서 사용하는 악의적인 노드를 검증하기 위한 복잡한 합의 알고리즘 등을 필요로 하지 않는다.

하이퍼레저 패브릭의 구성요소에는 크게 분산원장, 체인코드, Peer, Orderer 가 있다. 우선 블록체인의 핵심인 분산원장(Distribued Ledger)는 모든 로그 기록이 저장되어 있는 블록체인(Blockchain) 과 현재상태를 저장해놓은 월드 스테이트(World State)로 나뉜다. 노드의 종류는 보증피어(Endorsing peer), 커밋피어(Commiting Peer), 오더링 노드(Ordering node)로 구성된다.[15]

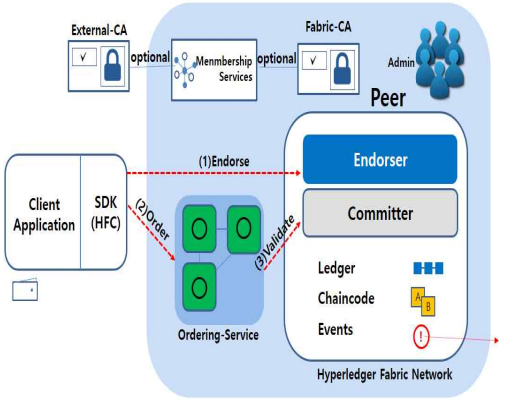



그림 2. 하이퍼렛저 패브릭 아키텍처

## 2.2 관련 연구

### 2.2.1 정보시스템 감리

정보시스템 감리에 대한 연구 주제는 크게 데이터모델링, 감리자동화, 데이터베이스보안, 데이터품질, 특정시스템에 대한 점검항목도출이 있다. 그러나 블록체인기술을 적용한 시스템을 위한 점검항목 연구는 아직 부재하다. 데이터모델링 관련한 연구에서는 데이터모델을 기준으로 정형화된 감리기준을 제시한 사례가 있고, 감리자동화 도구를 사용하여 SQL튜닝[4]과 데이터품질진단[5]관련한 연구가 있다. 이 논문에서는 자동화 도구를 사용하여 데이터베이스 구조를 분석하고 논리/물리데이터일관성, 데이터구조가 성능에 미치는영향을 조사하여 성능 개선방안에 대한 연구를 진행하였다. 데이터베이스 보안감리[6]관련한 연구에서는 데이터베이스 보안 감리 프레임워크를 제시하였고, 데이터품질을 향상하기 위해 DB오류유형분석및 도메인무결성을 분석하고 DB품질인증제도[7]를 제안한 논문도 있다. 점검항목도출에 관련한 연구로는 데이터품질의 적시성 확보를 위한 질의어(SQL)점검항목도출[8]에 관한 연구가 있다. 점검항목의 중요도를 분석하여 선택과 집중적으로 감리를 제시[9]하는 연구도 있다. 4년간 수행한 정보시스템 감리 프로젝트 및 위험발생 프로젝트 데이터를 중심으로 정보시스템구축 프로젝트의 품질향상을 위해 개선된 검토항목을 제시[10]한 연구도 있다.

### 2.2.2 블록체인기반 시스템개발 관련 연구

본 논문에서 제시하고자 하는 블록체인 기반의 시스템 개발 사업에 대한 정보시스템 감리에 관련한 연구는 부

재하나 블록체인 기반의 시스템개발 사업에 대한 연구로는 아키텍처설계, 개발방법론, 스마트컨트랙트 취약점과 관련한 연구가 있다. 좀더 구체적으로 말하자면, 기존개발방법론의 UML을 사용하여 블록체인 개발에 적용[11]한 연구가 있고, 블록체인 어플리케이션을 개발하기 위해 기민한(Agile) 소프트웨어공학방법을 제시[12]한 연구와 스마트컨트랙트의 취약점에 대한 대안을 제시[13]한 연구가 있다. 정보시스템감리와 관련된 연구는 아니지만 소프트웨어개발에 대한 연구이기 때문에 정보시스템감리 관련연구로 포함하였다.

## 3. 블록체인기반시스템에서 감사항목 추출

### 3.1 기존 감리점검항목의문제점

한국정보화진흥원에서는 기존의 감리체계가 정보화사업의 규모나 유형에 무관한 동일한 감리체계 운영으로 인한 비효율성이 존재하여 이를 개선하기 위한 방안 마련이 필요하다는 상황을 인식하여 2016년 정보시스템 규모.유형별 감리 참조 모델(안)을 개발 보고서에서 감리점검 프레임워크를 사업유형 기반 점검체계와 사업특성 기반 점검체계로 구성하였다.[17] 하지만, 2020년 현재 점검항목은 아직 구체화되지 않은 상태이다.

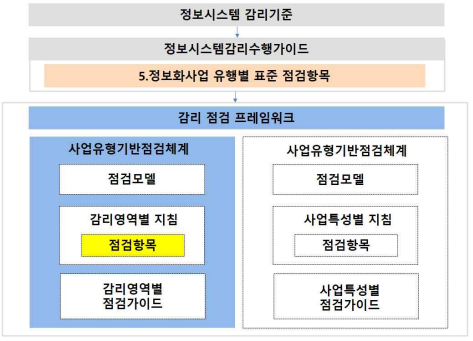



그림 3. 감리점검 프레임워크 개정안

이와 같이 정보시스템의 기술은 신속하게 변화하는데 비해 점검항목이나 가이드는 기술변화의 속도를 따라가지 못하고 있다. 또한, 기존 프레임워크의 점검항목이 얼마나 문제가 있는지에 대한 분석이 미흡하다. 따라서, 본 연구에서는 기존 점검함목이 블록체인과 같은 신흥기술에 적용시 얼마나 적합하지 않은가 분석하는 것부터 시작하였다.

시스템개발 사업유형에서 표준 감리점검항목은 아래와 같다. 개발방법론에 따라 2종류의 점검항목이 제시되어 있는데, 객체지향·컴포넌트기반 모델과 구조적·정보공학적 모델이다. 본 연구에서는 객체지향·컴포넌트기반 모델의 요구분석, 분석설계, 구현단계에서 점검사항을 대상으로 검토하였다. 감리영역은 응용시스템과 데이터베이스로 제한하였다. 응용시스템의 상세점검항목 수는 생명주기 단계인 요구분석, 분석설계, 구현단계별로 표 1과 같이 5개, 15개, 6개로 총 26개이다. 데이터베이스의 상세점검항목 수는 표 1과 같이 생명주기 단계인 요구분석, 분석설계, 구현단계별로 3개, 10개, 4개로 총17개이다.

표 1. 시스템개발- 감리점검항목 건수 (단위: 개)

| 모델 | 단계 | 시스템 구조 | 응용 | DB |
|---|---|---|---|---|
| 객체지향·컴포넌트기반모델 | 요구분석 | 4 | 5 | 3 |
| | 분석설계 | 5 | 15 | 10 |
| | 구현 | 3 | 6 | 4 |
| 구조적·정보공학적 모델 | 분석 | 5 | 5 | 8 |
| | 설계 | 4 | 6 | 8 |
| | 구현 | 3 | 6 | 4 |
| 시험활동 | | 9 | | |
| 운영준비 | | 4 | | |

### 3.2 델파이기법을 사용한 감사항목 도출

#### 3.2.1 델파이기법의 개요

델파이 기법은 명확한 증거가 없거나 거의 없고 의견이 중요한 사안에 대해 개인으로부터 집단적 견해를 얻는 방법이다. 그 과정은 그룹 소유권을 야기할 수 있고 다양한 관점을 가진 개인들 간의 결속을 가능하게 할 수 있다. 그것은 익명의 패넬리스트 그룹에 대한 통제되고 반복적인 피드백을 가진다.[15] 델파이 프로세스는 다른 곳에서 종합적으로 검토되었다(Adler & Ziglio, 1996; Skulmoski, Hartman, & Krhn).

#### 3.2.2 감사항목도출 절차

아래 그림은 본 논문에서 델파이 절차를 커스터마이징하여 사용된 감리점검항목 도출 절차다. 입력물로는 블록체인 경험, 하이퍼렛저 패브릭 기술서, 개발산출물이고, 결과물은 세번의 설문조사와 인터뷰를 통해 도출된 최종 점검항목이다.

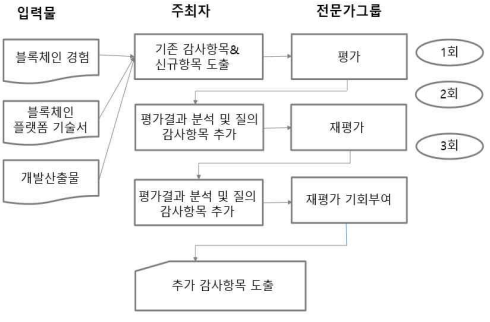



그림 4. 추가 감사점검항목 도출 절차

### 3.2.3 결과

3라운드 델파이 기법을 사용한 결과, 아래와 같은 새로운 감리점검프레임워크와 새로운 감사 항목이 도출되었다.

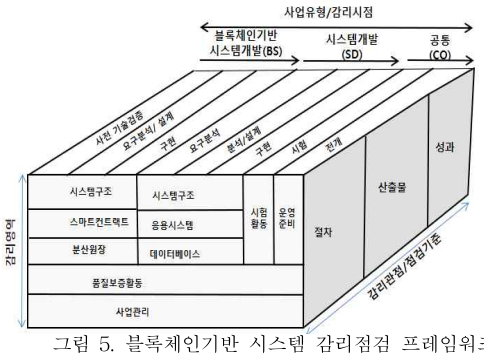



그림 5. 블록체인기반 시스템 감리점검 프레임워크

(1) Data 표준화 : Worldstate에 사용된 db의 형식이 산출물(eg. 테이블정의서, 컬럼정의서)로 작성되고, 표준화는 지켜야할 대상인지는 협의내용을 점검한다.

(2) 보안소스코드는 점검을 위한 TOOL은 GO 언어를 지원 아직 지원하지 않기때문에 전문가들이 직접 소스코드를 점검한다.

(3) 데이터무결성: 웹에서 입력된 값이 손상되지 않고 Ledger에 포함되는지 점검한다.

(4) 앵커링: 블록단위의 해쉬값과 블록모임에 대한 해쉬값이 제대로 생성되고 앵커링된 사이트(예:이더리움)에 저장된값과 내부에서 관리하는 해쉬값이 동일한 지 점검한다.

(5) 암호화 키값:개인키와 공개키가 있는 폴더가 외부에노출되지 않는지 점검한다.

(6) 개인정보가 원장에 저장되는지 점검한다. 개인정보에는 이메일,주민등록번호,여권번호,핸드폰번호가 있고 통신사에서 제공한 CI값도 개인을 추정할 수 있기 때문에 저장하는것은 적합하지 않다. World state DB에 개인정보가 저장되는지 점검한다.

(7) WORLD STATE : CouchDB/LevelDB의 값과 분산원장의 최종저장값, NON-DLT시스템에서 사용하는 RDB(관계형데이터베이스)의 컬럼과 매핑되고 데이터가 훼손되지 않는지 점검한다.

(8) 사용자Login 정보와 같이 개인정보보호법에 의해 삭제해야할 데이터가 분산원장에 포함되도록 설계되었는지 점검한다.

(9) 체인코드의 버전변경시, 데이터무결성 확보가능하도록 설계되었는지 점검한다. 체인코드 변경시 패브릭에서는 새로운 오브젝트로 인식하기 때문에 두개의 버전이 동시 사용되어 데이터를 오염시킬수 있다.

## 4. 새로 감리점검항목의 적정성 검증

### 4.1 설문개요

본 연구에서 추가된 감리점검 항목과 기존 감리점검항목의 적정성은 설문조사를 통해 확인하였다. 설문지 구성은 응답자의 블록체인기반 시스템에 대한 경험, 기존 감리프레임워크기존감리항목 적용의 적정성, 추가된 감리점검항목의 적정성으로 크게 3개로 분리하여 블록체인 기반 시스템 구축 감리 경험이 있는 정보시스템 감리원을 대상으로 설문조사를 실시했다.

설문조사는 20명에게 설문조사를 의뢰했고, 20명이 그 결과에 응답했다. 20명 모두 "일부 감리 상세점검항목은 커스터마이징 없이 사용할 수 있지만, 블록체인 기반 시스템에서도 새로운 항목이 추가돼야 한다"고 답해 본 논문은 연구가 필요하다고 결론지었다.

### 4.2 신뢰도 분석 및 적합도 분석

설문결과 20명이 응답하였고 아래의 조건에 따라 설문결과를 삭제하거나 감리점검항목에 포함 시켰다. 설문조사에 대한 신뢰도 분석을 위해 크롬바하 알파(Cronbach's alpha) 계수값을 산출하여 0.7이상의 값이 나와 신뢰성에 문제가 없다고 판단하였다.

Cronbach Alpha :

[ 항목의 개수 / (항목의 개수 -1) ] *

{ 1- [ SUM( var(xi) ) / var( SUM(xi) ) ] }를 적용한 결과이다. 이를 식으로 표하면, 아래와 같다.

k 는 항목의 개수, σ2yi 항목i에 대한 분산, σ2 는 전체점수에 대한 분산을 의미한다.

$$\alpha = \left(\frac{k}{k-1}\right)\left(1-\frac{\sum_{i=1}^{k}\sigma_{y_i}^2}{\sigma_x^2}\right)$$

감리점검항목의 적정성을 분석하기 위해 기술통계기법을 사용하였다. 그 결과는 아래 표 2.와 같다. 기존 감리점검항목이 적합이상의 비율은 적합항목의 비율인 35.4%와 매우적합을 받은 점검항목 16%를 합한 비율인 51.4%인데 비해 추가항목에 대한 적합이상의 비율은 적합항목 52%와 매우적합 31%를 합한 83%로 32.4%가 더 높게

표 2. 감리점검항목 적합도

| 구분 | 구분 | 매우부적합 | 부적합 | 보통 | 적합 | 매우적합 |
|---|---|---|---|---|---|---|
| 기존항목 | 점검항목건수 | 2 | 147 | 263 | 305 | 143 |
| | 비율% | 0.2% | 17.0% | 30.5% | 35.4% | 16% |
| 추가항목 | 점검항목건수 | 0 | 0 | 29 | 95 | 56 |
| | 비율% | 0 | 0 | 16% | 52% | 31% |
| **(기존-추가)항목 차이(%)** | | **-0.2%** | **-17%** | **13.5%** | **16.6%** | **19%** |

나왔다. 이는 추가점검항목이 좀 더 구체적으로 표현되어 적합이상의 평가가 더 많이 나온 것으로 판단된다. 매우 적합을 선택한 점검항목일수록 논란의 여지가 적은 감리점검항목이라고 볼 수 있다.

### 4.3 설문결과

기존 감리점검항목의 적정성여부에 대한 의견이 추가항목에 대한 편차는 표 2. 와 같이 32.4%차이 났다. 그이유는 기존점검항목에 대한 설명이 더 추상적이고, 명확하지 않아서 이런 결과가 나온 것으로 판단된다. 이는 한국정보화진흥원의 연구보고서에서 기존 점검항목의 문제점을 지적한 것과 같은 결과로 볼 수 있다.

## 5. 결론 및 향후 연구

정보시스템감리는 공공기관의 소프트웨어 개발시 의무사항이다. 따라서, 본 연구는 정보시스템 감리원은 물론이고 블록체인기반의 시스템개발을 하는 사업자와 발주자에게도 시스템의 안정적 효율적인 운용을 위해서 필요할 것이다.

향후 연구과제로는 현재는 패브릭기준으로 점검항목을 도출하였으나 이더리움 등 다른 블록체인 플랫폼에 대한 점검항목에 대한 연구가 필요하고 공통사항을 도출하여 플랫폼과 상관없이 공통적으로 점검할 수 있는 점검항목 및 점검기법에 대한 연구가 필요하다. 또한, 데이터표준화, 기능점수기반의 사업정보 산출, 블록체인에 소프트웨어보안 적용방법, 블록체인의 기능점수 산정법 등 시스템 개발사업 시 준수해야 할 지침을 블록체인에 적용하는 방안에 대한 연구가 필요하다.

마지막으로, 본 연구에서는 도출된 점검항목과 기법이 타당한지 체계적인 검증 프로세스에 대한 연구가 미흡하다. 체계화된 감리점검항목 도출을 위한 검증프로세스가 제시된다면 검증된 점검항목으로 감리를 수행함에 따라 정보시스템 감리에 대한 신뢰도가 높아질 것으로 예측된다.

## 참 고 문 헌

[1] 전자정부법,[법률 제12592호,2014.5.20.,타법개정]
[2] 한국정보화진흥원,“정보시스템감리수행가이드 v2.1”,2013.12.
[3] 한국정보화진흥원,“정보시스템감리수행가이드 v2.1”,p12, 2013.12.
[4] 김병철,"정보시스템 감리에서 데이터베이스 감리의 기준 연구",『한국콘텐츠학회 종합학술대회 논문집』,2권(1호),2004.,467-470
[5] 김창관,『데이터베이스 감리 개선 방안에 관한 사례연구』,동아대학교 경영대학원 학위논문(석사),2005.
[6] 김광열,『데이터 안전성 확보를 위한 데이터베이스 보안 감리점검 프레임워크 연구』,건국대학교 정보통신대학원 학위논문(석사),2008.
[7] 김종원,『데이터베이스 질의어 감리점검항목 도출을 통한 감리개선방안』,인천대학교 정보기술대학원 학위논문(석사),2013.
[8] 김활중,『정보시스템 감리를 통한 데이터베이스 품질향상』,건국대학교 정보통신대학원 학위논문(석사), 2011
[9] 이부형. (2013). AHP 기법을 이용한 정보시스템 감리 표준점검 항목의 선택 및 관리방법에 관한 연구. 한국정보기술학회논문지, 11(4), 177-184.
[10] 이돈희, 정홍섭, 이기영, 한기준. (2012). 정보시스템 감리사례 분석을 통한 품질 향상 방안에 관한 연구. 한국컴퓨터정보학회논문지, 17(10), 203-216.
[11] M. A. Awad, A Comparison between Agile and Traditional Software Development Methodologies, 2005
[12] Michele Marchesi, Lodovica Marchesi,Roberto Tonelli, An Agile Software Engineering Method to Design Blockchain Applications, arXiv:1809.09596,25 Sep 2018
[13] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli, A survey of attacks on Ethereum smart contracts, Universit_a degli Studi di Cagliari, Cagliari, Italy,
[14] Elli Androulaki Artem Barger Vita Bortnikov IBM, Srinivasan Muralidharan∗ State Street Corp.,Chet Murthy∗ Binh Nguyen∗ State Street Corp.,Manish Sethi Gari Singh Keith Smith Alessandro Sorniotti IBM, Chrysoula Stathakopoulou Marko Vukolić Sharon Weed Cocco Jason Yellick IBMHyperledger Fabric: A Distributed Operating System for Permissioned Blockchains, arxiv:1801.10228v2 [cs.DC], IBM,17 Apr2018
[15] https://www.slideshare.net/alehors/hyperledger-fabric-technical-deep-dive-20190618?from_action=save.io.
[16] Manlu Liu; Kean Wu;Jennifer Jie Xu , How Will Blockchain Technology Impact Auditing and Accounting: Permissionless versus Permissioned Blockchain, Current Issues in Auditing (2019) 13 (2): A19 - A29.https://doi.org/10.2308/ciia-52540
[17] 한국정보화진흥원, 정보시스템 규모·유형별 감리 참조모델(안) 개발, 2016년 12월

이 영 주
1995년 홍익대학교 경영정보학과 학사졸업
2007년 서강대 정보통신대학원 석사졸업
2013년 서강대학교 컴퓨터공학과 박사수료
2007~2018년 ㈜한국정보기술단 수석감리원
2018년~현재 서강대 지능형블록체인연구센터 재직중
관심분야: 정보시스템감리, 블록체인, 소프트웨어공학

박 수 용
1986년 서강대학교 컴퓨터공학과 졸업(학사)
1988년 Florida State University 컴퓨터및정보과학 석사
1995년 George Mason University 정보기술학 박사
1998년~現 서강대학교 컴퓨터공학과 교수
관심분야: 블록체인,요구공학,소프트웨어공학

# 키워드 기반 탐색적 테스트의 실험적 연구
## (Experimental Study of Keyword-Based Exploratory Testing)

황 준 선 ¶ 최 은 만 §
(Jun Sun Hwang) (Eun Man Choi)

**요 약** 탐색 테스트는 빠른 개발 주기라는 특징으로 바람직한 테스트 방법으로 소개되었으나 적용을 위하여 문서화 및 테스트 범위의 분석이 요구되어 적극적으로 채택하지 않고 있다. 한편 키워드 기반 테스트는 리소스 절약 및 유지 관리를 용이하게 하는 방법으로 소개되었으나 데이터, 설정, 상호 작용, 시퀀스 및 타이밍과 같은 변수가 많아 테스트를 미리 계획하는 것이 쉽지 않다. 하지만 키워드 기반 테스트에서 키워드를 작성하기 위한 명확한 기준과 방법을 제시하고 탐색 테스트 프로세스를 적용하여 키워드를 기반으로 테스트 사례를 만들 수 있다. 이 논문에서는 키워드 기반으로 탐색적 테스트를 자동화 하는 모델을 제안하고 실험한다. 효과를 검증하기 위해 일반 키워드 기반 테스트(KBT)와 탐색적 키워드 기반 테스트(KBET)와 비교하였고 탐색적 정상 테스트 사례(ETC) 및 탐색적 키워드 기반 테스트(KBET)와 비교하였다.

**키워드** : 탐색적 테스트, 키워드 기반 테스트, 테스트 자동화, 테스트 케이스 생성

**Abstract** The exploratory test was introduced as a desirable test method due to its fast development cycle, but it is not actively adopted because documentation and analysis of the test range are required for application. On the other hand, keyword-based testing has been introduced as a way to save resources and facilitate maintenance, but it is difficult to plan tests in advance due to the large number of variables such as data, settings, interactions, sequence and timing. However, in keyword-based testing, you can create a test case based on keywords by presenting clear criteria and methods for creating keywords and applying the exploration testing process. In this paper, we propose a model that automates exploratory tests based on keywords. To verify the effectiveness, we compared the general keyword-based test(KBT) and keyword-based exploratory test(KBET), and compared with the exploratory normal test case(ETC) and keyword-based exploratory test(KBET).

**Key words** : Exploratory testing, Keyword-based testing, Test automation, Test-case generation

## 1. 서 론

컴퓨터가 일상생활에 널리 사용되면서 다양한 응용 분야에 여러 가지 서비스를 제공하는 소프트웨어가 필요하고 개발할 때 빠른 출시와 중단 없는 패치와 핫 픽스 및 배포를 위한 기술이 일반화 되고 있다[1, 2]. 빠른 출시와 유지보수를 위하여 애자일 개발 프로세스가 소개되었고 더불어 대상 SW를 탐구하여 작은 테스트부터 시도하여 그 결과가 다음 테스트에 응용되는 탐색적 테스트(Exploratory Test: ET)가 주목을 받았다[7, 8, 9].

하지만 철저한 커버리지를 미리 분석하고 테스트 케이스를 준비하여 단계적으로 테스트하는 전통적인 프로세스에서는 탐색적 테스트는 채택하지 않아 왔다[16]. 그 이유는 테스트 케이스 생성 과정이 정형화 되어 있지 않고 리스크 기반의 테스트 세션(charter)을 실행하면서 학습하며 다음 테스트를 계획하고 기록하는 매우 휴리스틱한 방법이기 때문이다.

한편 키워드 기반 테스트(Keyword-based Test: KBT)는 테스트 대상 요소들에 대한 키워드를 정의하여 키워드 풀을 구성하고 동작, 설명, 데이터로 구성된 액티비티를 테이블 형태로 정의하고 액티비티의 시퀀스를 이용하여 테스트 케이스를 생성하기 때문에 정형적인 테스팅 방법이다. 이러한 특징으로 인하여 테스트 자원의 할당과 유지보수가 쉬울 뿐만 아니라 자동화를 가능하게 한다[17]. 키워드 기반 테스트는 빠른 개발 주기에서의 자원의 절약과 유지보수를 쉽게 할 수도 있다. 하지만 데이터, 설정, 상호작용, 순서, 시기 등 변수가 너무 많기에 모든 조건을 포함하는 테스트를 미리 계획하는 것은 부담이 될 수 있다.

¶ 일반회원 : ㈜아이오시스 연구원
hwangsun12@naver.com
§ 종신회원 : 동국대학교 컴퓨터공학과 교수
emchoi@dongguk.edu
논문접수 : 2020. 9. 28.
심사완료 : 2020.11.12

이 논문에서는 키워드 기반 테스트에서 키워드와 테스트 케이스를 작성하는 명확한 프로세스를 적용한 탐색적 테스팅을 제안하여 다음과 같은 문제를 해결하고자 한다. 첫째 휴리스틱한 탐색적 테스트를 키워드 기반으로 정형화 하고 자동화 하는 문제와 둘째, 키워드 기반 테스트에서 테스트 케이스를 작성하는 프로세스를 탐색적 테스트 개념을 적용하여 의미 있는 자동화 키워드를 작성하는 문제이다.

자동화된 탐색적 테스트를 지원하는 새로운 테스트 방식인 탐색적 키워드 기반 자동화 테스트(Keyword-Based Exploratory Test: KBET)에 대한 연구 결과를 확인하기 위하여 다음 두 가지 질문을 검증하는 실험을 설계하고 실행하였다. 먼저 '매뉴얼 기반 대신 탐색적 프로세스를 적용하여 키워드 기반 테스트를 진행하면 더 많은 오류를 탐지할 수 있는가?'와 두 번째로 '탐색적 키워드 기반으로 테스트 케이스를 작성하면 탐색적 일반 테스트 케이스보다 더 많은 오류를 탐지할 수 있는가?'이다. 이를 위하여 온라인 쇼핑몰과 항공예약 서비스를 대상으로 실험을 수행하였다.

본 논문의 나머지 구성은 2장에서 탐색적 테스트와 키워드 기반 테스트에 대한 배경을 설명하고 3장에서는 탐색적 키워드 기반 테스트 프로세스와 절차에 대해 자세히 설명한다. 4장에는 실험 설계와 분석된 결과를 기술하고 5장에 결론 향후 연구를 담았다.

## 2. 탐색적 테스트와 키워드 기반 테스트

### 2.1 탐색적 테스트

탐색적 테스트는 1983년 캠 카너에 의해 처음 소개되었고, 1987년 제임스 바크를 통해 널리 알려진 테스트 방식으로 테스터가 학습하면서 테스트 설계와 테스트 수행을 동시에 하는 것을 말한다. 탐색적 테스트는 테스트 케이스를 먼저 작성하지 않고 우선 테스트 대상 제품을 실행하면서 익숙해짐과 동시에 테스트를 설계하고 계획한다. 그로 인해 테스트 케이스 작성 시간을 최소화하고 테스터의 경험적인(Heuristic) 능력을 최대한 활용하는 방법이다.

탐색적 테스트의 특징은 수행될 각 세션에 테스터의 임무를 설정해 놓은 테스트 명령(테스트 차터)으로 제품의 리스크를 기반으로 작성된다. 즉 리스크가 높은 기능에 차터를 많이 생성하여 테스트 효율을 높인다. 또한 테스트에 몰입할 수 있도록 테스트 차터의 각 세션당 테스트 수행 시간을 60분, 90분, 120분 내외로 제한한다. 정형적인 테스트 케이스나 결과 보고서가 아니라 제품에 대한 기록, 발견된 결함과 이슈, 테스트한 방법이나 절차 등을 기술한 요약 등 가볍게 쓰는 테스트 노트를 사용한다.

결국 탐색적 테스트는 테스터의 경험에 의존하여 소프트웨어 제품을 직관적으로 테스트하는 방법을 제공한다. 하지만 경험이 적거나 새로운 SW를 테스트한다면 일반적으로 스크립트 테스트 접근법보다 더 많은 시간과 자원을 필요로 한다[16]. 이러한 이유로 탐색적 테스트를 일반적인 표준 테스트 접근법으로 사용하지 않는다[6, 7, 8]. 게다가 탐색적 테스트는 테스트 문서를 적게 생성하는 경향이 있다. 테스트 문서는 조직에서 테스트 실행 전 계획을 위해 필요하며[6, 7] 문서화 및 테스트 케이스는 테스트 팀의 성과와 진행 상황을 모니터링 하기 위하여 중요한 요소이다.

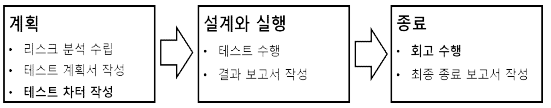



그림 1. 탐색적 테스트 작업 과정

### 2.2 키워드 기반 테스트

키워드 기반 테스트는 명세 기반 테스트 설계 기법을 이용하여 테스트를 수행하는 접근법으로 테스트 자동화 및 테스트 자동화 프레임워크 개발을 지원하는 일반적인 방식이다. 키워드는 컴퓨터 프로그래밍 언어가 아니라 테스트 케이스를 작성하는 데 사용하는 일반적인 추상화된 문장이다. 명세기반으로 설계한 매뉴얼 테스트 케이스의 수행 문장을 실행 가능한 키워드를 이용하여 SW 코드로 작성한 자동화 테스트 케이스라 할 수 있다. ISO 29119-5 키워드 기반 테스트 표준에서는 우선 매뉴얼 테스트 케이스를 작성하고 그것을 실행할 수 있는 키워드 테스트 프레임워크[2]를 제시하고 있다.

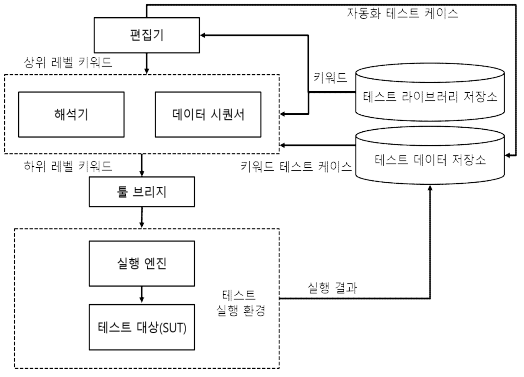



그림 2. 키워드 테스트 프레임워크 개념도

먼저 편집기를 이용하여 키워드 테스트 라이브러리의 키워드를 만들고 테스트 데이터로 그림 3과 같은 테스트 케이스를 작성한다. 다음은 키워드 레벨과 순서에 따라 해석 절차를 거치고, 키워드 자동화 실행 엔진을 통해 키워드 테스트 케이스가 실행된다. 이때 테스트 도구는 키워드 테스트 케이스가 자동으로 동작하도록 하는 역할을 한다. 키워드 기반 테스트 프레임워크에서 사용하는 데이터에는 키워드 테스트 라이브러리와 테스트 데이터, 키워

드 테스트 케이스가 있다. 테스트 저장소를 잘 구성하면 테스트 실행 결과와 로그도 한꺼번에 관리할 수 있는 특징이 있다.

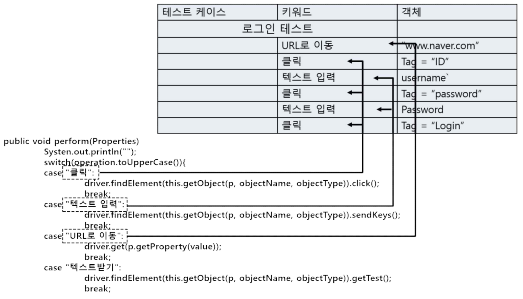



그림 3. 키워드 기반 테스트 케이스 작성

테스트 케이스의 실행은 Robot framework, kUTAF, Telerik Test Studio, Jenkins 등 다양한 테스트 자동화 도구와 연계하여 가능하다. 이중에 Robot framework는 오픈소스이며 파이썬 기반으로 확장성이 좋아서 다양한 외부 라이브러리를 지원하므로 키워드 기반의 테스트를 적용하기에 적합하다[6].

이와 같이 키워드 기반의 테스팅은 테스트 케이스를 구조화 하고 실행을 자동화 할 수 있는 좋은 특징들을 가지고 있다. 따라서 앞서 살펴본 팀색적 테스팅이 비구조적이며 휴스틱한 테스트 프로세스의 개선과 SW 품질 향상에 도움을 주는 보완적인 기술로 적합하다. 이점을 착안하여 Zylberman, Shenare의 연구[7]에서는 외부 녹화 도구를 사용하여 키워드를 파악하고 이를 기반으로 탐색적 테스팅을 자동화하는 가능성을 제시하였다. Hellmann et al.[1]의 연구에서는 탐색적 테스트를 키워드 기반으로 자동화하여 효율성을 높이는 가능성을 제시하였다. 그러나 위 연구들에서는 구체적인 프로세스의 제시와 도구를 사용하여 방법의 효과에 대한 검증이 없다. 본 논문에서는 KBET에 대한 구체적인 프로세스를 제안하고 그 효과성 검증을 위한 실험을 하였다. 또한 KBET를 지원하는 새로운 프레임워크도 설계하였다.

## 3. 키워드 기반 탐색적 테스트 자동화 방법

본 연구의 핵심인 키워드 기반 탐색적 테스트의 자동화 접근 모델은 그림 4와 같이 6단계 절차로 진행한다. 각 절차는 이전 절차의 산출물이 사용되기 때문에 순서대로 진행해야 한다.

KBET 프로세스를 수행을 위한 절차로 먼저 매뉴얼 기반으로 테스트 케이스를 작성한다. 그 다음 작성한 테스트 케이스에서 키워드를 추출한다. 다음 탐색적 테스팅을 수행하고 탐지한 에러를 기반으로 테스트 케이스 추가, 키워드 리팩토링을 해준다. 마지막으로 테스트를 수행하고 결과를 분석하는 과정으로 진행된다. 이후 3~6 과정을 반복하며 테스트 케이스를 추가하고, 키워드를 리팩토링 해준다. 만약 한 개발주기가 끝나고 업데이트로 새로운 기능이 생겼을 경우에는 1~6 과정을 다시 수행해준다.

그림 4의 2단계에서 키워드 추출은 Robot Framework의 자동화 소스코드를 작성하는 과정이다. 2단계에서 추출한 키워드로 작성한 테스트 케이스는 테스트를 수행하는 단계인 그림 4의 6단계에서 테스트 수행과 리포트 생성 자동화가 이루어진다.

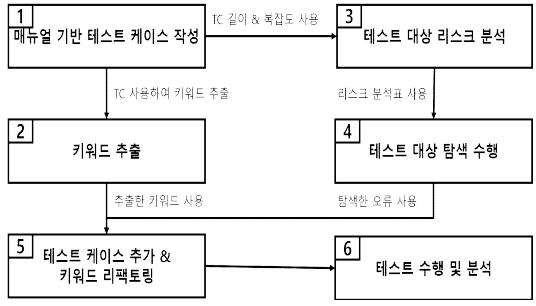



그림 4. 키워드 기반 탐색적 테스트 프로세스

### 3.1 매뉴얼 기반 테스트 케이스 작성

매뉴얼 기반 테스트 케이스를 작성할 때는 해당하는 기능이 동작이 되는지 여부만 판단 한다. 데이터, 설정, 상호작용, 순서, 시기 등 변수가 너무 많기에 모든 조건을 포함하는 테스트를 미리 계획하는 것은 불가능하기 때문이다. 매뉴얼 기반 테스트 케이스는 그림 5와 같이 [로그인 페이지 클릭 → id 입력 → pw 입력 → 로그인 버튼 클릭 → 올바른 로그인 페이지가 나오는지 확인] 절차로 테스트 케이스를 작성할 수 있다.

'버튼 클릭', '텍스트 입력' 등 간단한 동작을 수행하는 키워드를 하위 레벨 키워드라고 하는데, 하위 레벨 키워드를 조합하여 로그인, 주문하기 등 작은 업무 단위를 상위 레벨 키워드로 작성할 수 있고, 상위 레벨 키워드를 또 조합하여 더 큰 단위의 상위 레벨 키워드를 작성할 수 있다. 키워드로 작성한 테스트 케이스는 Robot framework를 사용하여 매크로처럼 업무를 실제 동작시키면서 테스트를 수행한다.

그림 5의 테스트 케이스는 가장 왼쪽에 있는 것이 하위 레벨 키워드이고 나머지는 매개변수이다. Click Link는 "/login" 이라는 링크를 가지고 있는 요소를 클릭한다. Wait Until Element Is Visible은 클릭 이후 "id=name"

| | | | |
|---|---|---|---|
| 1 | SeleniumLibrary.Click Link | /login | |
| 2 | Wait Until Element Is Visible | id=name | |
| 3 | Input Text | id=name | userid |
| 4 | Input Password | id=password | userpw |
| 5 | Click Button | name=commit | |
| 6 | Element Should Be Visible | id=Products | |
| 7 | Sleep | 1s | |

그림 5. TC_2 Login을 테스트하기 위한 소스 코드

이라는 경로가 확인 될 때까지 기다린다. Input Text는 "id=name" 이라는 경로에 "userid"를 입력한다. Element Should Be Visible은 "id=Products"라는 경로가 보이면 테스트를 Pass 아니면 Fail 처리한다.

온라인 쇼핑몰 SW 제품에는 로그인, 로그아웃, 계정 삭제, 계정 등록, 상품 등록, 상품 수정, 상품 삭제 등의 기능이 있다고 할 때 요구사항 목록을 식별하여 총 13가지의 테스트 케이스를 도출할 수 있다. 하지만 장바구니 관리의 제품 추가와 제품 관리는 카테고리 설정이 3가지가 있어 각각 3개씩 테스트 케이스를 만들어 주었다. 자동화된 테스트 케이스를 재사용하였을 때 문제가 없도록 추가한 제품 삭제도 3가지를 만들어 준다.

## 3.2 키워드 추출

키워드 추출을 하면 기존 자동화 테스트 케이스의 유지 보수가 쉬워지며 이후 테스트 케이스를 확장할 때 훨씬 낮은 비용으로 확장할 수 있게 된다.

그림 6은 매뉴얼로 작성한 인터넷 쇼핑몰의 제품 편집 자동화 테스트 케이스를 키워드로 추출하기 위해 매개변수 값을 바꾸어 주었다. 각 파라미터 값을 변수 바꾸어 주어 키워드로 추출하였다. 다음과 같이 매뉴얼로 작성된 테스트 케이스를 변수 값들만 변경해주면 마치 함수처럼 자동화 테스트 케이스를 쉽게 재사용할 수 있다. 변경한 파라미터는 ${fixed_title}, ${description}, ${price}이다.

| | | | |
|---|---|---|---|
| 1 | Click Element | xpath=//tr[@id='${target_title}']//a[contains(text(),'Edit')] | |
| 2 | Wait Until Element Is Visible | xpath=//input[@id='product_title'] | |
| 3 | Input Text | xpath=//input[@id='product_title'] | ${fixed_title} |
| 4 | Input Text | xpath=//textarea[@id='product_description'] | ${description} |
| 5 | Click Element | xpath=//select[@id='product_prod_type'] | |
| 6 | Wait Until Element Is Visible | xpath=//option[contains(text(),'Books')] | |
| 7 | Click Element | xpath=//option[contains(text(),'${prodType}')] | |
| 8 | Input Text | xpath=//input[@id='product_price'] | ${price} |
| 9 | Click Element | xpath=//input[@name='commit'] | |
| 10 | Wait Until Element Is Visible | xpath=//p[@id='notice'] | 3 |
| 11 | Sleep | 1s | |

그림 6. 매개변수를 변경한 Edit Product 테스트 케이스

키워드는 테스트 자동화의 핵심적인 역할을 한다. 키워드의 이름이나 내용 측면에서 상세한 정도, 구조에 따라 테스트 케이스 구성이 영향을 받는다. 특히 키워드 이름은 이를 사용하는 사람들이 자연스럽게 떠올릴 수 있는 단어나 문장을 이용하여 정의하는 것이 중요하다.

키워드는 상위 레벨 키워드와 하위 레벨 키워드로 구성된다. 상위레벨 키워드는 입력 매개변수와 하위 레벨 키워드로 구성되고 각 키워드는 식별할 수 있도록 의미 있는 이름을 부여한다. 그림 7의 Edit Product라는 상위 레벨 키워드는 그림 6의 테스트 케이스를 추출한 키워드로 마치 함수처럼 사용이 가능하다. 키워드 추출은 Robot Framework 도구에서 지원해준다. 그림 7의 테스트 케이스는 로그인을 하고, 제품 수정을 자동으로 수행하는 테스트 케이스이다. 추출한 Edit Product 키워드를 왼쪽에 작성하고, 오른쪽에는 제품명, 수정할 제품명, 제품 설명, 카테고리, 수량 등의 파라미터를 입력하여 키워드를 사용할 수 있다. 키워드로 작성한 테스트 케이스는 유지보수하기에 적합하다. 또한 수없이 많은 조합 테스트를 키워드와 파라미터 입력만으로 자동 수행이 가능하다.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | Login | TestAccount | testtest | | | |
| 2 | Edit Product | prodname | Fixed prodname | Fixed product description | Other | 10 |

그림 7. 키워드를 사용하여 작성한 Edit Product 상위 테스트 케이스

## 3.3 테스트 대상 리스크 분석

리스크 분석은 일반적으로 [장애 발생 가능성 × 장애로 인한 영향도]로 계산한다. 테스트 대상에 따라 리스크 요소는 변동될 수 있다. 리스크 요소는 이해관계자들과의 회의를 통해 정하고 가중치를 설정한다. 테스트 난이도는 테스트 케이스의 길이, 복잡도는 사용자가 입력하는 파라미터의 개수, 상호관계는 연관된 모듈의 수로 계산하였다. 장애로 인한 영향도는 직접 판단하여 설정하였다. 그림 8은 테스트 대상을 리스크 분석한 결과이다. 그림 9는 그림 8의 결과를 바탕으로 [장애 발생 가능성 × 장애로 인한 영향도]를 계산한 결과이다.

| 리스크 요소 / 리스크 아이템 | 장애 발생 가능성 | | | 장애로 인한 영향도 | | |
|---|---|---|---|---|---|---|
| | 테스트 난이도 | 복잡도 | 상호 관계 (인터페이스) | 사용 빈도 | 경제적 피해 | 사용자에 의한 취급 중요도 |
| 계정 등록 | 1 | 1 | 3 | 1 | 9 | 3 |
| 시스템 로그인 | 1 | 1 | 3 | 3 | 9 | 3 |
| 시스템 로그아웃 | 1 | 1 | 3 | 3 | 9 | 3 |
| 계정 삭제 | 1 | 1 | 3 | 1 | 5 | 3 |
| 제품 추가 | 5 | 5 | 5 | 5 | 5 | 5 |
| 제품 편집 | 5 | 5 | 5 | 5 | 5 | 5 |
| 제품 삭제 | 5 | 5 | 5 | 5 | 5 | 5 |

그림 8. 테스트 대상 리스크 분석

## 3.4 테스트 대상 탐색

분석한 리스크를 기반으로 테스트 대상을 탐색한다. 30분이라는 제한 시간 동안 위험요소를 우선순위로 두어 수행을 한다. 탐색적 소프트웨어 테스팅은 테스터 개개인의 자유 의지와 책임감을 강조하는 소프트웨어 테스팅의 한 방법으로 테스터마다 탐색 방법이 다르다. 하지만 주관적인 차이를 줄이기 위하여 테스트 실험 대상에서 탐색을 수행할 때 기능 테스트, 이상 값 입력, html 소스 분석, UI 확인 등의 테스트를 대상으로 한다. 탐색 후 그림 9와 같은 4개의 오류를 탐지할 수 있다.

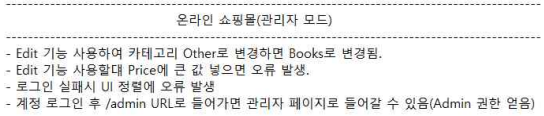



그림 9. 테스트 대상 탐색을 통해 탐지한 오류

## 3.5 테스트 케이스 추가 및 키워드 리팩토링

탐색한 오류가 기존 테스트 케이스의 커버리지에 포함되지 않는 경우에는 테스트 케이스를 추가한다. 오류가 테스트 커버리지에 포함되지만, 오류를 탐지하지 못할 경우에는 키워드를 리팩토링 한다. 예를 들어 커버리지에 포함되지 않는 경우에 추가된 테스트 케이스는 다음과 같다.

1) Editing Product를 수행할 때 Price 값에 큰 값을 넣으면 오류 발생
2) 로그인할 때 실패하면 UI 정렬에 오류가 발생한다.
3) 계정 로그인 후 /admin URL을 입력하면 관리자 권한을 얻는다.

탐지한 3가지 오류를 위한 테스트 케이스를 추가하여 키워드를 정의하면 그림 10과 같다.

| | | | |
|---|---|---|---|
| 1 | Click Element | xpath=//tr[@id='${target_title}']//a[contains(text(),'Edit')] | |
| 2 | Wait Until Element Is Visible | xpath=//input[@id='product_title'] | |
| 3 | Input Text | xpath=//input[@id='product_title'] | ${fixed_title} |
| 4 | Input Text | xpath=//textarea[@id='product_description'] | ${description} |
| 5 | Click Element | xpath=//select[@id='product_prod_type'] | |
| 6 | Wait Until Element Is Visible | xpath=//option[contains(text(),'Books')] | |
| 7 | Click Element | xpath=//option[contains(text(),'${prodType}')] | |
| 8 | Input Text | xpath=//input[@id='product_price'] | ${price} |
| 9 | Click Element | xpath=//input[@name='commit'] | |
| 10 | Wait Until Element Is Visible | xpath=//p[@id='notice'] | 3 |
| 11 | Page Should Contain | ${fixed_title} | |
| 12 | Page Should Contain | ${description} | |
| 13 | Page Should Contain | ${prodType} | |
| 14 | Page Should Contain | ${price} | |
| 15 | Sleep | 1s | |

그림 10. 'Edit product' 테스트 케이스 리팩토링 결과

## 3.6 테스트 수행 및 분석

테스트 케이스를 추가하여 키워드 리팩토링이 완료되면 자동화된 테스트 케이스를 실행한다. 그림 11은 수행한 테스트 케이스의 결과이다. 왼쪽의 녹색 표시된 것은 통과한 테스트 케이스이다. 빨간색으로 표시된 것은 실패한 테스트 케이스이다.

여기에서 이상한 점 한 가지를 발견할 수 있다. 우리가 탐색하고 추가하고 리팩토링한 테스트 케이스는 TC_7(Edit test case1), TC_14(Login error test), TC_15(add product test4), TC_16(Admin access test)이다. 그런데 TC_9에서 추가로 오류가 탐지된 것을 확인할 수 있다. 이런 경우 테스트 로그를 확인하여 테스트 케이스가 실패한 원인을 분석할 수 있다.

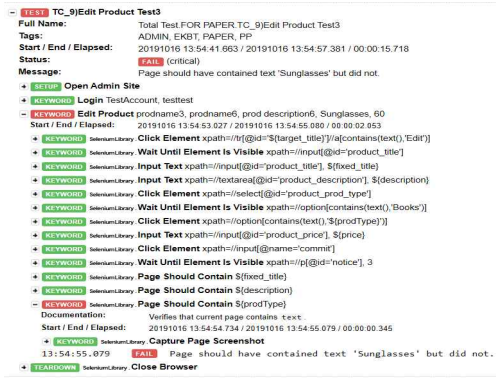



그림 11. 테스트 케이스 TC_9 테스트 로그

그림 11에서 빨간색으로 된 실패한 테스트 케이스인 아래에서 두 번째 테스트 케이스에서 "Page should have contained text 'Sunglasses' but did not."을 확인할 수 있다. 즉, 카테고리를 Sunglasses로 수정하였을 때 카테고리가 정상적으로 수정되지 않았음을 볼 수 있다. 앞장에서 리팩토링한 Edit Product 키워드가 새로운 오류를 검출해낸 것이다.

## 4. 실험과 검증

3장에서는 제안하는 방안인 KBET 프로세스 절차에 대해 자세히 설명하였다. 이번 장에서는 제안하는 KBET 모델을 검증하기 위하여 연구 질문을 세우고 실험을 수행하였다. 실험은 온라인 쇼핑몰과 항공 예약 서비스를 대상으로 하였다. 각각 사용자 모드와 관리자 모드가 존재한다. 즉 총 4가지 서비스를 대상으로 4번의 실험을 수행하였다. 실험 참여자는 저자 1명으로 제품의 업무 학습으로 인한 테스트 결과에 영향을 미치는 것을 방지하기 위하여 4번의 실험에 KBT, KBET, ETC 방법의 순서를 번갈아 가며 수행하였다.

온라인 쇼핑몰의 기능은 사용자 모드는 상품 주문, 결제, 목록 조회, 검색, 장바구니 관리, 카테고리별로 정렬 등의 기능이 있다. 관리자 모드는 로그인, 로그아웃, 계정 삭제, 계정 등록, 상품 관리 기능 등이 존재한다.

또한 항공 예약 서비스의 기능은 사용자 모드는 상품 목록 조회, 계정 등록, 로그아웃, 로그인, 계정 정보 수정, 위시리스트 조회, 상품 예약 등의 기능이 있다. 관리자 모드는 계정 관리, 쿠폰 관리, 대시보드 조회, 상품 등록, 로그인 등의 기능이 있다.

## 4.1 실험 방법

연구 질문 1: 매뉴얼 기반 대신 탐색적 프로세스를 적용하여 키워드 기반 테스트를 진행하면 더 많은 에러를 탐지하는가?

위의 연구 질문을 검증하기 위해 실험을 수행하였다. 검증을 위해 각각 4개의 시스템에서 각 90분씩 실험을 수행하였다. 매뉴얼 키워드 기반 자동화는 10분간 매뉴얼을 작성하고 80분간 테스트 케이스를 작성하는 방식으로 진행하였다. 제안 방안인 KBET는 리스크 분석 30분, 분석한 리스크 기반으로 테스트 대상 탐색 30분, 발견한 오류로 테스트 케이스 작성 30분씩 총 90분 동안 수행한다. KBET가 동일 시간 오류를 많이 찾는지 확인하기 위해 프로세스의 3~6 과정을 수행하였다.

연구 질문 2: 탐색적 키워드 기반으로 테스트 케이스를 작성하면 탐색적 일반 테스트 케이스보다 더 많은 에러를 탐지하는가?

연구 질문 2를 검증하기 위하여 탐색적 키워드 기반 자동화는 3장에서 설명한 KBET 프로세스를 동일하게 수행하였다. 탐색적 일반 테스트 케이스(ETC)는 KEBT 프로세스를 동일하게 수행하지만 탐지한 오류가 커버리지에 포함될 경우에 키워드 수정 대신 테스트 케이스를 수정하는 방법으로 진행한다.

## 4.2 리스크 분석

제안한 실험을 수행하기 위해 시스템별로 리스크 분석을 수행하였다. 리스크 분석에 대한 방법은 3 장에서 자세히 설명한 방법이다. 그림 12(a) 는 온라인 쇼핑몰에 대한 리스크 분석표이다. 관리자 모드와 사용자 모드에 대한 리스크로 각각 30분씩 리스크 분석을 수행하였다. 항공 예약 서비스에 대한 리스크 분석표이다. 항공 예약 서비스(그림 12(b))의 경우에는 30분이라는 리스크 분석 시간에 제한적인 서비스만 리스크 분석을 수행할 수 있었다. 또한 탐색 시간과 테스트 케이스 작성 시간도 제한적이라 일부 기능에 대해서만 리스크 분석을 하였다.

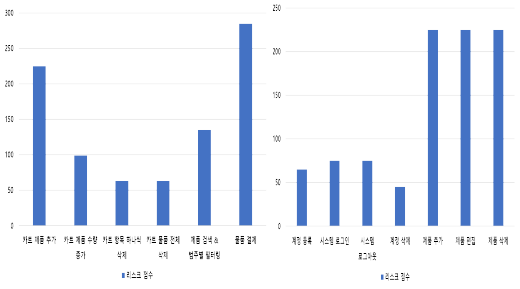
| 리스크 요소 / 리스크 아이템 | 장애 발생 가능성 | | | 장애로 인한 영향도 | | |
|---|---|---|---|---|---|---|
| | 테스트 난이도 | 복잡도 | 상호 관계 (인터페이스) | 사용 빈도 | 경제적 피해 | 사용자에 의한 취급 중요도 |
| 계정 등록 | 1 | 1 | 3 | 1 | 9 | 3 |
| 시스템 로그인 | 1 | 1 | 3 | 3 | 9 | 3 |
| 시스템 로그아웃 | 1 | 1 | 3 | 3 | 9 | 3 |
| 계정 삭제 | 1 | 1 | 3 | 1 | 5 | 3 |
| 제품 추가 | 5 | 5 | 5 | 5 | 5 | 5 |
| 제품 편집 | 5 | 5 | 5 | 5 | 5 | 5 |
| 제품 삭제 | 5 | 5 | 5 | 5 | 5 | 5 |
| 카트 제품 추가 | 5 | 5 | 5 | 5 | 5 | 5 |
| 카트 제품 수량 증가 | 3 | 3 | 3 | 3 | 3 | 5 |
| 카트 항목 하나씩 삭제 | 1 | 1 | 5 | 3 | 3 | 3 |
| 카트 물품 전체 삭제 | 1 | 1 | 5 | 3 | 3 | 3 |
| 제품 검색 & 범주별 필터링 | 5 | 5 | 5 | 3 | 3 | 3 |
| 물품 결제 | 5 | 5 | 5 | 5 | 9 | 5 |

(a) 온라인 쇼핑몰 리스크 분석표

| 리스크 요소 / 리스크 아이템 | 장애 발생 가능성 | | 장애로 인한 영향도 | |
|---|---|---|---|---|
| | 복잡도 | 연관 모듈 | 영향도 | 사용빈도 |
| 계정관리 | 9 | 3 | 5 | 3 |
| 쿠폰관리 | 3 | 3 | 5 | 3 |
| 대쉬보드 조회 | 3 | 5 | 5 | 5 |
| 상품 등록 | 9 | 5 | 5 | 5 |
| 로그인 | 5 | 3 | 5 | 3 |
| 상품예약 | 3 | 3 | 5 | 5 |
| 로그인/로그아웃 | 1 | 5 | 5 | 5 |
| 계정 등록 | 3 | 3 | 5 | 1 |
| 위시리스트 조회 | 1 | 3 | 3 | 3 |
| 계정 정보 수정 | 5 | 3 | 3 | 1 |
| 상품 목록 조회 | 5 | 9 | 5 | 9 |

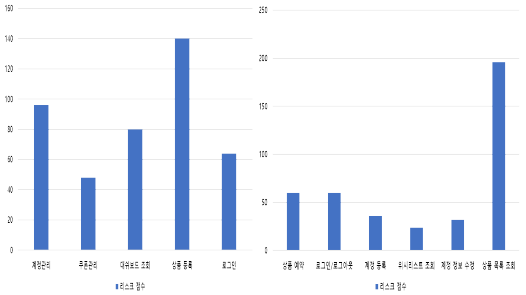
(b) 항공 예약 서비스 리스크 분석표

그림 12. 리스크 분석표



(a) 사용자 모드 (b) 관리자 모드

그림 13. 온라인 쇼핑몰 리스크 분석 결과



(a) 사용자 모드 (b) 관리자 모드

그림 14. 항공권 예약 시스템 리스크 분석 결과

## 4.3 탐색

그림 15는 평가한 리스크를 기반으로 각 30분씩 탐색을 수행하여 탐지한 오류이다. 리스크가 낮은 부분은 기능 테스트와 간단한 이상 값 입력 정도만 수행하고 리스크가 높은 부분은 기능 테스트, 이상 값 입력, html 소스 분석, UI 확인 등 다양한 테스트를 수행하였다. 특정 요소에서 오류가 발견될 경우에는 테스팅의 일반적인 원리 -4 결합 집중에 따라 요소에 대해서 우선순위를 높여 테스트를 수행하였다.

## 4.3 테스트케이스 작성 및 키워드 리팩토링

연구 질문 1을 검증하기 위한 첫 번째 실험에서는 KBT는 총 62개의 테스트 케이스를 작성하였고 KBET는 17개의 테스트 케이스를 작성하였다. KBT는 80분의 시간 동안 테스트 케이스를 작성하여 KBET보다 비교적 많은 테스트 케이스를 작성하였다. KBET는 많은 시간을 리스크 분석과 오류 탐지에 소요하여 더 적은 테스트 케이스를 작성하였다.

실험 2는 61개의 동일한 기능 테스트 케이스를 만든 상태에서 프로세스를 수행하였다. 이후 탐색한 오류를 추가하여 ETC는 총 21개의 테스트 케이스를 추가하였고 KBET는 16개의 테스트 케이스를 추가하고 5개의 키워드를 리팩토링하였다.

### 4.4 실험 결과

연구 질문 1을 검증하기 위한 첫 번째 실험 결과 KEBT는 온라인 스토어에서 7개, 항공 예약에서 8개 총 15개의 오류를 탐지하였다. KBT는 온라인 스토에어서 2개, 항공 예약에서 1개 총 3개의 오류를 탐지하였다. 동일 시간 테스트를 수행하였을 때 KBET가 KBT보다 5배 더 많은 오류를 탐지하였다.

연구 질문 2를 검증하기 위한 두 번째 실험 결과 KBET는 온라인 스토에어서 11개, 항공 예약에서 8개 총 19개의 오류를 탐지하였다. ETC는 온라인 스토어에서 10개, 항공 예약에서 6개 총 16개의 오류를 탐지하였다. KBET는 키워드를 리팩토링하여 탐색적으로 기존에 탐지하지 못했던 오류를 3개 더 탐지하였다.

## 5. 결론 및 향후 연구

본 논문에서는 탐색적 키워드 기반 자동화 테스트 모델 설계 및 검증을 하였다. 탐색적 테스트와 키워드 기반 테스트를 혼합한 모델들 제안함으로써 업계에서 탐색적 테스팅을 채택하지 않는 이유인 문서화와 커버리지 충족을 해결하였다. 또한 구체화되어 있지 않던 키워드 기반 테스트의 프로세스를 구체화하였다. 제안한 모델을 검증을 위한 연구 질문에 대한 실험 결과는 다음과 같다.

매뉴얼 기반 대신 키워드 기반 탐색적 테스트를 진행하면 더 많은 에러를 탐지하는가(연구 질문 1)에 대한 실험은 90분이라는 제한적인 테스트 자원을 주어 실험을 진행하였다. KBET는 필요한 것만 자동화며 자동화 비용을 줄여 동일 시간 동안 4개의 실험 대상으로 5배 많은 오류를 찾아낸다는 것을 확인하였다.

탐색적 키워드 기반으로 테스트 케이스를 작성하면 일반 탐색적 테스트 케이스보다 더 많은 에러를 탐지하는가(연구 질문 2)에 대한 실험은 키워드 리팩토링과 탐색을 통해 탐지하지 못한 오류를 추가로 찾아낼 수 있다는 것을 확인하였다. KBET 접근법을 통해 ETC보다 13% 더 많은 오류를 탐지하였다.

제안한 KBET 접근법을 통해 기존 KBT 방식과 ETC 방식보다 더 많은 에러를 탐지할 수 있다는 결론이 나왔다. 제안 방안은 개발 주기에서 제한적인 테스트 자원이 주어질 때 KBT, ETC보다 더 효율적인 사용이 가능하다고 예측된다. 이 외에 Zylbeman과 Shenare의 Automated Exploratory 연구에서 외부 녹화 도구를 사용하여 탐색적 테스트를 키워드 기반으로 자동화하는 가능성을 제시하였었다. 해당 논문을 참고하여 이전 연구에서 RAKTA라는 도구를 설계하여 동작 녹화를 통해 키워드로 자동화할 수 있도록 하였었다. 그러나 제안하는 절차를 사용하여 키워드를 잘 구성할 경우 녹화 도구는 불필요하다는 결론을 얻었다.

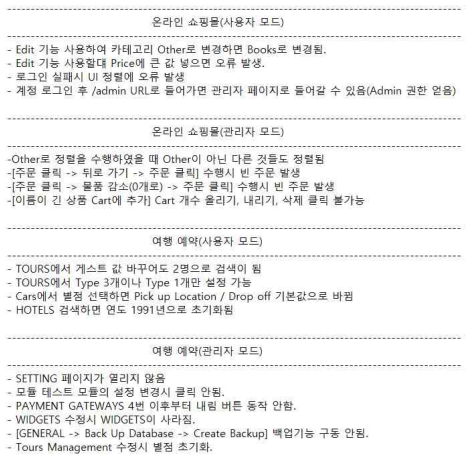



그림 15. 테스트 대상에서 탐색을 통해 탐지한 오류

본 실험은 제한적인 시간 내에 테스트 자동화와 많은 오류를 찾는 데에 핵심이 있다. 따라서 실무에 적용할 경우 아래 사항을 고려해야한다. (1) KBT를 사용하여 자동화한 더 많은 자동화 테스트 케이스를 장기적으로 회귀 테스트에 사용할 경우 KBET는 KBT보다 오류 탐지율이 떨어질 수 있다. (2) EBT를 사용하여 문서화 없이 테스트만 수행할 경우 KEBT는 EBT보다 오류 탐지율이 떨어질 수 있다.

또한 해당 실험은 적은 실험 횟수와, 적은 실험 참여자로 실험적 제한이 있다, 향후 실무에 적용하여 추가적인 실험과 함께 KBET가 장기적으로 효율적일지에 대한 검증이 필요하다.

## 참 고 문 헌

[1] 정보통신산업진흥원, “SW 공학 백서”, 2018.
[2] 정상미, “더 괜찮은 QA가 되기 위한 프렉티컬 테스트 자동화”, 2018.
[3] Y. S Lee and Y. M. Ha, "Software Testing by a keyword driven test automation method and Effects", 2005 NuriMedia Co., pp. 604~606. 2005.
[4] Y. Hwang, S. Jung, C. Hwa, "A Keyword-based UI Test Framework for Web Services", 정보과학회논문지: 소프트웨어 및 응용 제 38권 제 12호,

pp.657~662, 2011.
[5] J. Oh, S. Kim, J. Hwang, "Using Specification By Example and Keyword-based Test Automation for Agile Testing", 한국 소프트웨어 공학 학술대회 논문집 제 15권 제 1호, pp.428~433, 2013.
[6] E. Choi, M. Zhang, "Analysis and Improvement of Keyword-driven Auto-Testing Process Based on Robot Framework", 한국정보과학회 학술발표논문집, Vol. 45, No. 1. 2018.
[7] A. Zylberman and N. Shenar, "Automated exploratory testing," http://www.testingexcellence.com/automated-exploratory-testing-2, Feb. 2010.
[8] J. A. Whittaker, Exploratory Software Testing: Tips, Tricks, Tours and Techniques to Guide Test Design. Indianapolis: Addison-Wesley, 2010.
[9] J. Bach, "Exploratory testing explained," http://www.satisfice. com/articles/et-article.pdf, 2003.
[10] K. Li and M. Wu, Effective Software Test Automation: Developing an Automated Software Testing Tool. San Francisco: Sybex, 2004.
[11] E. Dustin, Effective Software Testing: 50 specific ways to improve your testing. New York: Addison-Wesley, 2003.
[12] E. Dustin, T. Garrett, and B. Guaf, Implementing Automated Software Testing: How to Save Time and Lower Costs while raising quality, 1st ed. Indianapolis: Addison-Wesley, 2009.
[13] A. Bacioccola, M. Catelani, L. Ciani, and V. L. Scarano, "Software automated testing: A solution to maximize the test plan coverage and to increase software reliability and quality in use," Computer Standards &Interfaces, pp. 152-158, Feb. 2011.
[14] M. Kelly, "Choosing a test automation framework," http://www.ibm.com/developerworks/rational/library/591.html, Nov. 2003.
[15] F. Bouquet, C. Grandpierre, B. Legeard, F. Peureux, N. Vacelet, and M. Utting, "A subset of precise uml for model-based testing," in Int'l. Workshop A-MOST, Jul., pp. 95-104, 2007.
[16] C. J. Schaefer, H. Do, "Model-Based Exploratory Testing: A Controlled Experiment", IEEE, April, 2014.
[17] ISTQB, "Certified Tester Foundation Level Syllabus", 2018.
[18] 권원일 외, "위험천만 테스팅", 2012.

황 준 선
2020년 2월 동국대학교 컴퓨터공학과 대학원 석사, 2020년 2월~현재 ㈜ 아이오시스 연구원, 관심분야는 소프트웨어공학, 소프트웨어 테스팅 자동화, 프론트엔드, 백엔드 개발,

최 은 만
1982 동국대학교 전자계산학과 졸업(학사) 1985 한국과학기술원 전산학과 졸업(공학석사) 1993 미국 Illinois Institute of Technology 전산학과 졸업(전산학박사) 1985 한국표준연구소 연구원 1988 한국데이타통신(주) 주임연구원 1997~2004 한국정보과학회 S/W 공학 연구회 운영위원 2001~2005 한국정보처리학회 학회지 편집위원 2000 / 2007 콜로라도 주립대(포트콜린스) 전산학과 방문교수 2002 카네기 멜론 대학 S/W 공학 단기 과정 연수 2003~2007 TTA 테스트 엔지니어 인력 양성 프로그램 운영 2014 베일러 대학 컴퓨터 과학과 방문교수 1993~현재 동국대학교 컴퓨터공학과 교수 관심분야는 객체지향 및 컴포넌트 S/W 공학, S/W 테스팅, S/W 품질 메 트릭, Aspect-Oriented Programming, Program Comprehension

# 논문지 논문 모집 (Call for Papers)

한국정보과학회 소프트웨어공학 소사이어티에서는 매년 2회에 걸쳐 '소프트웨어공학 소사이어티 논문지'를 발간하고 있습니다. 이 논문지에는 소프트웨어공학 전반에 걸친 연구 성과 및 산업계 동향을 소개하고 있습니다. 이에 다음과 같은 소프트웨어공학 주제에 관련된 논문을 모집하고 있으니 학계와 산업계의 여러분들의 적극적인 논문투고를 바랍니다.

◈ 논문 주제

- ○ 소프트웨어 설계, 아키텍처 및 프로덕트라인
- ○ 요구공학
- ○ 소프트웨어 품질 및 테스트
- ○ 프로젝트 관리 및 프로세스
- ○ 소프트웨어 정형 기법 및 검증
- ○ 임베디드, 모바일, 웹기반 소프트웨어 개발
- ○ 기타 소프트웨어 응용 (국방, 자동차, 의료, 조선, IoT, 빅데이터 등의 분야)

◈ 논문심사

- ○ 투고된 논문은 편집위원회에서 심사 선정하며, 필요 시 외부 심사위원을 위촉하여 심사를 합니다. 제출된 논문은 반환하지 않습니다.
- ○ 심사료 및 게재료: 없음

◈ 논문 제출

- ○ 한국정보과학회 논문지 투고 양식(www.kiise.or.kr)을 사용하며, 논문의 분량은 10장으로 제한합니다.
- ○ Journal of Software Engineering Society 논문투고시스템을 이용하여 제출바랍니다. (https://acoms.kisti.re.kr/journal/intro.do?page=logo&journalSeq=J000159)

◈ 문의처 (편집위원회)

- ○ 편집위원장 : 최윤자 교수 (경북대학교, 053-950-7549, yuchoi76@knu.ac.kr)
- ○ 편집이사 : 지은경 교수 (한국과학기술원, 042-350-7810, ekjee@se.kaist.ac.kr)
- ○ 편집이사 : 이주용 교수 (울산과학기술원, 052-217-2123, jooyong@unist.ac.kr)

# 투 고 요 령

1. 소프트웨어공학소사이어티 논문지에 실리는 원고는 주제 논문, 일반 논문, 산업체 기고 등으로 구분하며 다음과 같은 분야에 대하여 모집한다.
   가. 소프트웨어공학 및 그 응용분야에 대한 연구결과
   나. 강좌 및 관련 교육사항 소개 (목적, 과정, 일정, 대상, 특징)
   다. 소프트웨어 도구 및 방법론 소개 (가격, 특징, 종류, 적용사례)
   라. 소프트웨어 산업에 대한 학계, 업계의 주요 관심사
   마. 기타 관련 사항

2. 투고자는 원칙적으로 본 소사이어티의 회원으로 한다. 다만 공동 또는 초청 기고자는 예외로 한다.

3. 논문은 원칙적으로 한글로 작성한다.

4. 원고는 한글(hwp), 워드(MS Word), PDF 형식 중 하나를 택하여 작성하며, 그림과 표를 포함하여 10쪽 이내로 한다.

5. 논문 내용에 직접 관련이 있는 문헌에 대해서는 이들 문헌에 관련이 있는 본문 중에 참고 문헌 번호를 쓰고 그 문헌을 참고문헌 난에 인용 순서대로 기술한다. 참고문헌은 학술지의 경우 저자, 제목, 학술지명, 권, 호, 쪽수, 발행 연도의 순서로, 단행본은 저자, 서명, 쪽수, 발행처, 발행 연도의 순서로 기술한다.

6. 기타 자세한 사항은 한국정보과학회 논문지 투고 요령을 따른다.

## 소프트웨어공학소사이어티 논문지 제29권 제2호 (통권 106호)

발 행 일 ॥ 2020년 11월 30일

발 행 인 ॥ 홍장의

편 집 인 ॥ 최윤자

발 행 처 ॥ 사단법인 한국정보과학회 소프트웨어공학소사이어티

연 락 처 ॥ 충북 청주시 서원구 충대로 1번지 충북대학교
전자정보대학 3관 320호 소프트웨어학과 홍장의
전 화 : 043-261-2261, 팩 스 : 043-273-2265
홈페이지 : http://www.sigsoft.or.kr/

# Journal of Software Engineering Society

VOLUME 29, NUMBER 2, November 2020



Korean Institute of Information Scientists and Engineers

Software Engineering Society