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.

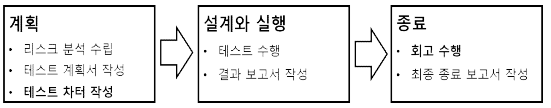


그림 1. 탐색적 테스트 작업 과정

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.

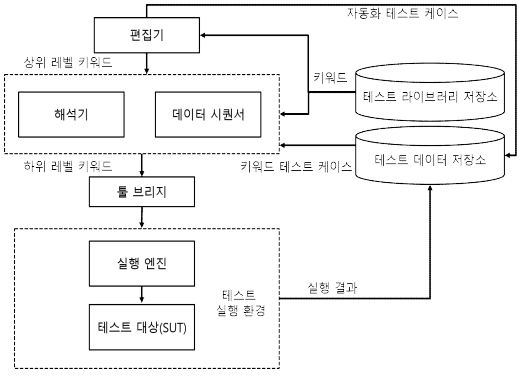


그림 2. 키워드 테스트 프레임워크 개념도

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.

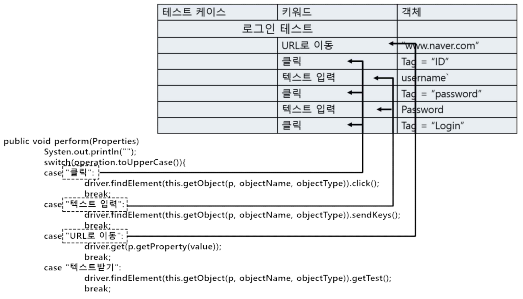


그림 3. 키워드 기반 테스트 케이스 작성

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.

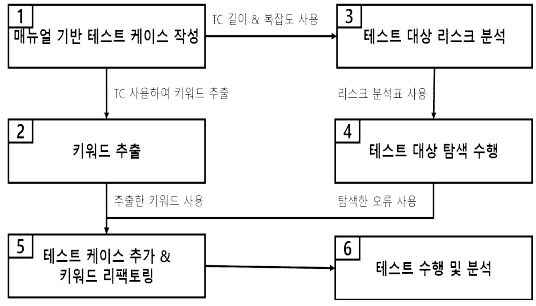


그림 4. 키워드 기반 탐색적 테스트 프로세스

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.

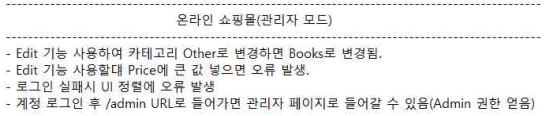


그림 9. 테스트 대상 탐색을 통해 탐지한 오류

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.

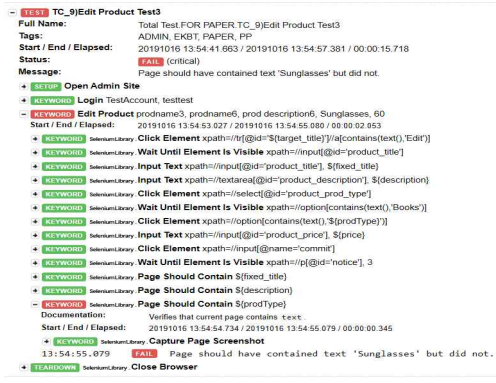


그림 11. 테스트 케이스 TC_9 테스트 로그

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.

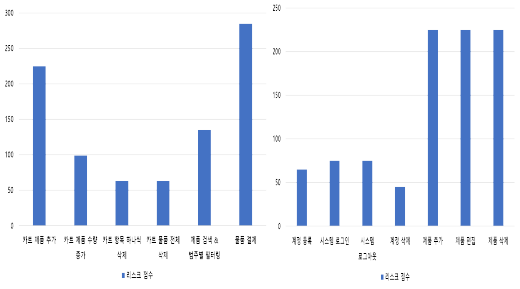
리스크 요소 / 리스크 아이템	장애 발생 가능성			장애로 인한 영향도		
	테스트 난이도	복잡도	상호 관계 (인터페이스)	사용 빈도	경제적 피해	사용자에 의한 취급 중요도
계정 등록	1	1	3	1	9	3
시스템 로그인	1	1	3	3	9	3
시스템 로그아웃	1	1	3	3	9	3
계정 삭제	1	1	3	1	5	3
제품 추가	5	5	5	5	5	5
제품 편집	5	5	5	5	5	5
제품 삭제	5	5	5	5	5	5
카트 제품 추가	5	5	5	5	5	5
카트 제품 수량 증가	3	3	3	3	3	5
카트 항목 하나씩 삭제	1	1	5	3	3	3
카트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

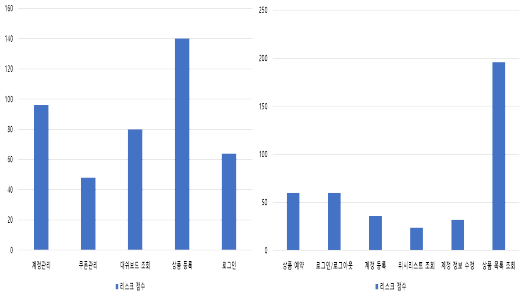
(b) 항공 예약 서비스 리스크 분석표

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.

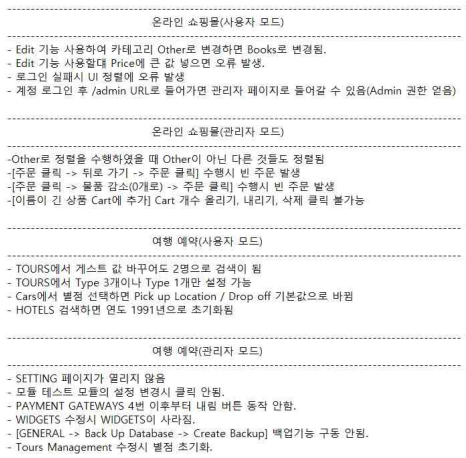


그림 15. 테스트 대상에서 탐색을 통해 탐지한 오류

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