韓國IT서비스學會誌
第17卷 第1號
2018年 3月, pp.155-170

Journal of Information Technology Services
https://doi.org/10.9716/KITS.2018.17.1.155

데이터 주도 접근법을 활용한 소프트웨어 테스트 자동화 : 온라인 쇼핑몰 결제시스템 사례

김성용* · 민대환** · 임성택***

Software Test Automation Using Data-Driven Approach : A Case Study on the Payment System for Online Shopping

Sungyong Kim* · Daihwan Min** · Seongtaek Rim***

■ Abstract ■

This study examines a data-driven approach for software test automation at an online shopping site. Online shopping sites typically change prices dynamically, offer various discounts or coupons, and provide diverse delivery and payment options such as electronic fund transfer, credit cards, mobile payments (KakaoPay, NaverPay, SyrupPay, ApplePay, SamsungPay, etc.) and so on. As a result, they have to test numerous combinations of possible customer choices continuously and repetitively. The total number of test cases is almost 584 billion. This requires somehow automation of tests in settling payments.

However, the record playback approach has difficulties in maintaining automation scripts due to frequent changes and complicated component identification. In contrast, the data-driven approach minimizes changes in scripts and component identification. This study shows that the data-driven approach to test automation is more effective than the traditional record playback method. In 2014 before the test automation, the monthly average defects were 5.6 during the test and 12.5 during operation. In 2015 after the test automation, the monthly average defects were 9.4 during the test and 2.8 during operation. The comparison of live defects and detected errors during the test shows statistically significant differences before and after introducing the test automation using the data-driven approach.

Keyword : Software Test Automation, Regression Test, GUI Automation Test, Data-Driven Approach, Online Payment System

Submitted : November 5, 2017 1st Revision : December 23, 2017 Accepted : December 26, 2017

* 고려대학교 일반대학원 디지털경영학과 박사과정, ㈜넥슨, 주저자
** 고려대학교 경영정보/디지털경영학과 교수, 교신저자
*** 고려대학교 경영정보/디지털경영학과 교수

1. 서 론

인터넷의 생활화와 정보통신기기의 활발한 사용으로 전자거래는 점점 더 증가하고 있다(Hwang and Jeong, 2016; Jo et al., 2013). 특히 제품과 서비스를 구매하는 대금 지불이 실시간으로 이루어지면서 전자결제는 전자거래의 핵심적인 요소가 되었다(Bolt et al., 2003).

전자거래가 이루어지는 온라인 쇼핑몰은 무통장 결제, 카드결제, 실시간 계좌이체, 가상 계좌이체, 휴대폰결제, 적립금 및 예치금 결제 등 다양한 결제 방식을 사용하고 있다. 또한 최근에는 여러 가지 간편결제 서비스(페이코, 페이나우, 카카오페이, 케이페이, 네이버페이, 시럽페이)가 등장하면서 빠르게 온라인 쇼핑몰시장에 확대되었다(Kim et al., 2017; Jin et al., 2017). 이용률이 가장 높은 네이버페이는 2015년 6월에 서비스를 시작하여 11개월 만에 가입자 1,500만 명, 가맹점 약 10만 개, 누적 거래액 1조 원을 돌파하였고, 2014년 9월에 시작된 카카오페이는 24개월 만에 가입자 1,000만 명, 가맹점 약 1,000개, 누적 거래액 1조 원을 돌파하였다. 페이코는 2015년 8월 이후 17개월 만에 가입자 500만 명, 가맹점 약 20만 개, 누적 거래액 1조 원을 넘어섰고, 페이나우는 2013년 11월에 출시하여 가입자 435만 명, 가맹점 약 11만 개, 케이페이는 2014년 12월에 시작하여 가입자 300만 명, 가맹점 약 10만 개, 시럽페이는 2015년 4월에 시작하여 가입자 250만 명, 가맹점 약 6만 개를 확보하였다(DMC Media, 2016).

이처럼 다양한 결제수단이 제공되면서 쿠폰, 카드할인, 포인트 등 결제금액을 계산하는 과정에서 크고 작은 변동사항이 수시로 발생하여 온라인 쇼핑몰은 결제 프로그램을 지속적이고 반복적으로 테스트해야 한다. 이런 환경에서 테스트 효과를 극대화할 수 있는 자동화 테스트 방법이 요구되고 있다.

테스트 주도 개발(Test-Driven Development) 방법론을 기반으로 JUnit, CUnit, PHPUnit 등을 이용한 단위 테스트 자동화(Damm and Lundberg, 2006), 또는 Record-Playback 기법을 이용한 회귀 테스트(Regression Test) 자동화 구현 사례들이 연구되고 있다(Meszaros et al., 2003; Memon et al., 2005). 하지만, 단위테스트 자동화는 개발자의 역량에 따라 테스트 수행 결과에 대한 편차가 심하고, 프로그램 원시코드와 테스트 코드를 함께 작성해야 하는 개발자의 부담이 크며, 테스트 코드와 원시코드 사이에 결합도가 높아 테스트 케이스에 대한 수정도 어렵다는 문제점이 있다. 또한, 작성된 테스트 코드는 단위 및 통합 테스트에만 활용되고, 시스템 테스트나 인수 테스트에서는 활용되지 못한다(Polo et al., 2007).

Record-Playback 기법을 이용한 회귀테스트 자동화는 솔루션의 잦은 업데이트에 따라 자동화 스크립트도 함께 수정해야 하는 유지보수와 녹화된 수많은 케이스의 자동화 스크립트 실행에 많은 시간이 소요된다. 이를 분산 실행하기 위해서는 가상 실행 환경에 많은 자원을 투입해야 하므로 실무에 바로 활용하는 데는 많은 어려움이 있다(Memon et al., 2005; Zech et al., 2017).

온라인 쇼핑몰의 결제과정에서 회원정보, 상품정보, 결제수단 등 다양한 조합에 따라 생성되는 수십만 가지의 테스트 케이스를 수동 테스트로 실행하는 것은 불가능하므로, 본 연구에서는 기존의 테스트 자동화 방법들을 분석하여 도출한 Record-Playback 기법과 데이터 주도 접근법(Data-Driven Approach)을 통합한 회귀테스트 자동화 방식의 효과를 입증하고자 한다. 나아가 제시된 테스트 자동화 방법은 온라인 쇼핑몰뿐만 아니라 특정 시나리오 안에서 여러 테스트 조건을 조합하여 생성된 테스트 케이스를 실행하는 사업 환경(포털 검색, 전자결재 시스템, 예매 또는 예약 시스템)에서도 응용하여 적용하는 것이 가능할 것으로 여겨진다.

본 논문의 전체 구성은 다음과 같다. 제 1장 서론에 이어, 제 2장에서 회귀 테스트와 GUI(Graphic User Interface) 테스트 자동화 방법론과 제약사항, 데이터 주도 접근 방식에 대한 선행연구를 살펴본다. 제 3장에서 결제 정합성 자동화의 설계 방법과 테스트 자동화 구조를 소개하고, 제 4장에서는 테스트

자동화 적용 사례의 결과를 설명한다. 마지막 제 5장에서는 본 연구의 의의와 한계를 기술하였다.

2. 이론적 배경

2.1 회귀 테스트(Regression Test)

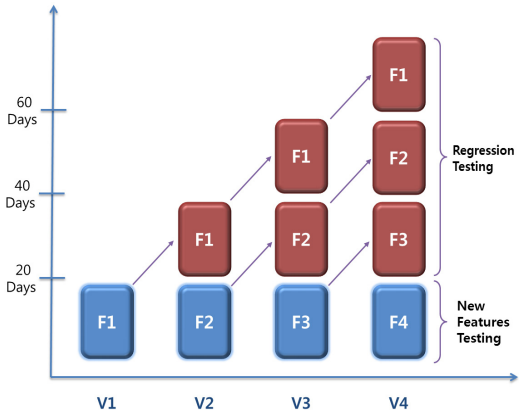
회귀 테스트란 개발이 완료된 소프트웨어에 대한 요구사항 변경, 오류 수정, 업데이트 등으로 새로운 버전의 소프트웨어를 개발할 때, 변경된 기능으로 인해 기존 기능에 오류가 발생하지 않는지, 또는 예상치 못한 오류의 유입은 없는지 확인하고, 소프트웨어의 다른 부분에 영향을 미치는지 여부를 테스트 하는 것이다(Kwon and Lee, 2010).

회귀 테스트가 반복적으로 수행되는 이유는 발견된 결함이 정확히 수정되었더라도 의도하지 않았던 새로운 형태의 결함이 발생하거나, 프로그램 소스코드의 수정 또는 소프트웨어 모듈의 추가 과정에서 기존에 정상 동작했던 기능에 악영향을 끼쳐 결함이나 오류를 발생시키지 않는지를 재검증하기 위한 목적으로 수행된다(Jeong et al., 2008).

이러한 회귀 테스트는 모든 테스트 수준(단위, 통합, 시스템, 인수 테스트)에서 수행할 수 있으며, 반복적인 성향을 가지고 있어 많은 비용과 시간이 증가한다는 단점이 있다(Rothermel et al., 2004; Schwartz and Do, 2016).

<Figure 1>은 개발된 소프트웨어 모듈이 변경되고 추가됨에 따라 회귀 테스트의 검증 범위가 지속적으로 증가하고, 결국에는 전체 범위로 확대된다는 것을 보여주고 있다. <Figure 1>과 같이 소프트웨어 버전이 증가될 때마다 신규 기능은 F1 > F2 > F3 > F4순으로 추가되지만, 회귀테스트 검증 범위는 F1 > F1, F2 > F1, F2, F3으로 급증하기 때문에 시간과 비용도 크게 증가하게 된다.

이러한 문제를 해결하고, 회귀테스트의 효율성을 높이기 위해 테스트 항목을 재사용하는 다양한 연구가 진행 중이며, 여러 방법론이 나오고 있다. 대표적인 방법론으로는 Retest-All, Regression Test Selection, Test Suite Reduction, Test Case Prioritization 등이 있으며, 각 기법들에 대해 간단히 소개하면 아래와 같다(Rothermel et al., 2004).



〈Figure 1〉 Regression Testing Scope
(Source : ProtoTech Solutions)

1. **Retest-all** : 이전에 사용했던 모든 테스트 케이스를 새로운 버전에 모두 재사용하는 것으로 비용과 노력이 많이 소요되어 매우 비효율적이지만, 수정된 코드로 인한 오류를 발견할 확률이 높다. 예를 들면, A기능 테스트를 위해 100개의 테스트 케이스로 테스트 했다면, A+기능으로 수정된 경우에도 기존의 100개의 테스트 케이스를 모두 테스트 수행하고, A+에 대한 추가 테스트 케이스를 수행한다.
2. **Regression Test Selections** : 소스가 변경된 영역을 분석하여 변경된 영역과 연관된 테스트 케이스를 선택적으로 재사용하는 하는 것이다. 연관된 테스트 케이스를 많이 선택할 수 록 검증의 신뢰성을 높일 수 있다. 예를 들면 A+기능으로 수정된 경우 기존의 100개의 테스트 케이스 중 변경된 기능과 연관된 50개 테스트 케이스만 선택하여 수행한다.
3. **Test Suite Reduction** : 향후 필요 없을 것으로 판단되는 불필요한 테스트 케이스를 영구적으로 삭제하고, 실제 테스트 스위트의 사이즈를 최소화하여 테스트하는 방식이다. RTS(Regression Test Selections)는 테스트 케이스를 일시

적으로 배제하는 기법이지만 TSR(Test Suite Reduction)은 테스트 케이스 삭제를 통해 테스트 스위트의 사이즈를 줄이는 기법이다. 테스트 케이스 삭제로 결함 검출 효율성이 저하될 가능성이 존재하지만 전체 재 테스트 기법에 비해 효과적이다.

4. **TCP(Test Case Prioritization)** : 모든 테스트 케이스를 특정 기준으로 우선순위를 매겨 우선순위가 높은 순서대로 테스트 케이스를 선택하여 순차적으로 수행하는 기법이다. TCP는 짧은 시간 내에 테스트 커버리지를 높이는데 효율적이며, 위 세 가지 기법들에 비해 비용과 시간이 절감되는 것으로 나타났다(Elbaum et al., 2000; Do and Rothermel, 2006). 우선순위를 정하는 기준은 기능의 복잡도, 고객의 주요 요구사항, 오류가 가장 많이 발생하는 영역, 잦은 변경이 필요한 기능, 사용 빈도가 높은 영역 등의 기준으로 테스트 케이스의 우선순위를 지정할 수 있다(Jung et al., 2015).

위 소개된 기법들의 특징을 정리하면 <Table 1>과 같다.

회귀테스트는 검증 범위가 증가하는 문제와 함께 소프트웨어의 변경으로 인해 구현된 검증 대상의 기능과 기존에 작성한 테스트 케이스인 검증 항목간의 불일치로 많은 시간과 비용을 허비하게 된다. 변경된 기능이 테스트 케이스를 통해 정확히 검증되지 않거나, 변경된 검증 대상 기능의 테스트 케이스 자체가 누락된다면, 소프트웨어에 내포된 결함을 찾아 수정하는 것은 매우 어렵다. 이처럼 회귀테스트에서 검증 항목인 테스트 케이스는 가장 중요한 요소이며, 언제나 변경된 소프트웨어의 기능을 테스트할 수 있는 최신의 테스트 케이스가 유지될 수 있도록 관리되어야 한다.

위와 같이 다양한 회귀테스트 방법론이 존재하지만 회귀테스트의 모든 문제를 해결해 주지 못한다. 하지만 회귀테스트 방법론과 테스트 자동화 도구 및 방법론을 적절히 결합하면, 제품의 신뢰성과 테스트 효과성, 효율성을 높일 수 있는 좋은 방안이 될 것이다(Memon et al., 2005).

2.2 GUI 테스트 자동화 방법론과 제약사항

GUI 테스트는 사용자가 마우스 조작 또는 키보드 입력과 같은 이벤트를 통해 시스템과 상호작용 하는 것을 자동화 스크립트를 작성해 실행하면서 시스템의 오류나 결함을 검출하는 것이다. GUI 테스트에 사용되는 방법론은 Record Play Back, 스크립트 기반 테스트, 키워드 기반 테스트, 모델 기반 테스트, 데이터 기반 테스트 등이 있다(Lee et al., 2007).

GUI 테스트 자동화 방법론의 장단점을 간단히 정리하면 아래와 같다.

1. **Record Play Back** : 테스트 수행 행위를 자동으로 기록하고, 기록된 결과를 재실행해 동일한 기능의 테스트 케이스를 다시 수행할 수 있는 장점이 있다. 하지만, 기능 명세가 정확하지 않고, 코드 단위로 기능이 구현되지 않아 기능의 커버리지(Coverage)가 불명확하고, GUI 수정이 많은 경우 회귀테스트 수행 시 자동기록을 재작성 해야 하는 등, 스크립트 유지보수에 많은 비용이 발생한다.

〈Table 1〉 Regression Testing Techniques

	Focus	Cost Effectiveness	Removing Test Case	Prioritizing Test Case
Retest-all	Execution of whole test cases	Worst	X	X
Regression Test Selection	Selection of test cases	Good	X	X
Test Suite Reduction	Minimization of test suite size	Good	O	X
TCP	Prioritizing test cases	Best	X	O

2. **스크립트 기반 테스트** : 테스트 케이스를 자동으로 실행하기 위해 자동화 툴을 통해 UI의 속성 정보를 이용하여 스크립트를 작성하고, 생성된 자동화 스크립트를 반복적으로 실행할 수 있는 장점이 있다. 하지만, 기능의 커버리지가 불명확하고, 요구사항을 분석하는데 많은 시간과 비용이 투입된다. 또한, 복잡한 스크립트를 작성하는데 어려움이 따른다.
3. **키워드 기반 테스트** : 기능 자동화 테스트 프레임워크의 유형으로 테스트 케이스의 각 동작을 키워드로 정의한다. 정의된 키워드를 이용하여 시나리오 형태의 테스트 케이스를 생성하고, 작성된 테스트 케이스를 실행한다. 키워드 기반 테스트의 장점은 키워드 정보를 통해 프로그래밍을 접하지 않은 사용자들도 테스트 스크립트를 쉽고 유연하게 작성할 수 있다. 하지만, 기능의 커버리지가 불명확하고, 요구사항 분석에 많은 비용과 노력이 필요하다.
4. **모델 기반 테스트** : 소프트웨어 테스트를 수행하기 위해 설계하는 과정과 필요한 프로그램을 실행하는 과정에 모델 기반 설계 기법을 적용하는 것으로 시스템에서 요구하는 동작(desired behavior)을 모델로 표현하고 이를 기반으로 시스템의 요구사항을 테스트하는 것이다(Chae, 2014). 모델 기반 테스트의 장점은 새로운 기능이 추가될 때마다 테스트 케이스를 작성하지 않고, 모델을 통해 유의미한 테스트 케이스 생성이 가능하고, 단순히 코드 경로(Code Path)나 의존성(Dependency)이 바뀐 것을 검증하기 위한 회귀 테스트뿐만 아니라 테스트 범위를 확장해 오류를 발견해 내는 것도 가능해 테스트 커버리지 수준이 높다. 하지만, 기능의 커버리지가 불명확하고, 요구사항 분석에 많은 비용이 필요하며, 각각의 모델을 만들기 위해 대규모의 사전 작업이 필요하다.
5. **데이터 기반 테스트** : 데이터 풀, Excel 파일, ADO 개체, CVS 파일, ODBC 소스 등의 테스트 데이터 저장소에 테스트 대상 데이터를 생성하여 이를 이용해 테스트 하는 기법이다. 데이터 기반 테스트의 장점은 효율적으로 다양한 테스트 데이터 생성이 가능하고, 다양한 데이터 조합의 변경을 통해 동일한 테스트가 가능하다. 하지만, 기능의 커버리지가 불명확하고, 테스트 데이터 생성을 위한 분석에 많은 시간과 비용 투자가 필요하다.

위에서 설명한 GUI 테스트 자동화 기법은 다음과 같은 제약사항을 가지고 있다.

1. GUI 프로그램은 이벤트를 통해 사용자가 화면상에서 픽셀을 클릭하거나 광범위한 범위에서 동작을 취할 수 있어, 초점 이동의 예측이 어렵고, 초점의 이동 경로가 다양하고 복잡하다(Memon et al., 2005).
2. GUI 요소들은 어플리케이션 내부에서 GUI 요소들 사이에 상호 의존성을 가지고 동시에 복잡한 이벤트를 수행한다.
3. 화면의 픽셀 단위마다 마우스 이벤트와 다양한 입력 조건의 키보드 이벤트는 무수히 많은 입력과 출력이 존재하기 때문에 그 결과를 검증하는 것이 어렵다(Gerrard, 1997).
4. GUI 구성요소를 식별하기 위해 웹 페이지 구조를 코드 수준으로 파악해야 하며, 웹 페이지의 구성이나 시나리오 변경으로 인해 유지보수 비용이 많이 든다(Morris, 1999).

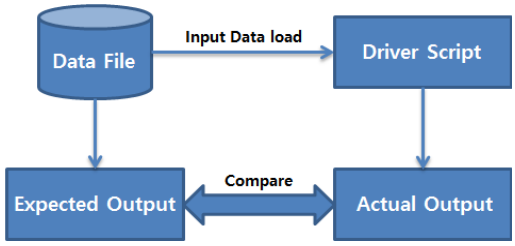
2.3 데이터 주도 접근 방식

데이터 주도 접근방식은 테스트가 실행될 때마다 동일한 특정 고정 값을 사용하는 것이 아니라 테스트 데이터와 출력 결과 값을 데이터 파일에서 읽어 들여 테스트 스크립트를 만드는 것이다.

일반적으로 테스트 입력 값과 예상 결과를 스프레드시트에 저장하고, 입력 데이터를 기준으로 실행결과와 예상결과를 비교하여 테스트를 수행한다(Kwon et al., 2010). 이러한 데이터 주도 접근법의 장점을 설명하면 다음과 같다.

1. 데이터 주도 테스트를 통해 테스트 코드 양을 최소화하면서 테스트 커버리지를 높일 수 있다.
2. 많은 양의 테스트 항목을 쉽게 만들어 실행할 수 있다.
3. 스크립트 언어에 익숙하지 않은 테스터라도 사전에 정의된 스크립트에 테스트 데이터를 입력하고 테스트를 수행할 수 있다.
4. 응용프로그램을 다뤄보지 않아도 테스트에 필요한 데이터를 구성하고 준비할 수 있다.

<Figure 2>는 이러한 장점을 가진 데이터 주도 접근법의 흐름을 다이어그램을 통해 표현한 것이다.



〈Figure 2〉 Flow Diagram of Data-Driven Approach (Source : w3ii.com)

본 연구에서는 GUI 테스트 자동화 문제를 해결하기 위해 Selenium을 이용한 Record Play Back 기법과 데이터 주도 접근법을 접목하여 자동화 스크립트는 최소화 하고, 다양한 데이터의 조합 조건을 통해 생성된 테스트 케이스를 효과적으로 실행하는 회귀테스트 자동화 구현 방안을 제시하고자 한다.

3. 결제 정합성 자동화

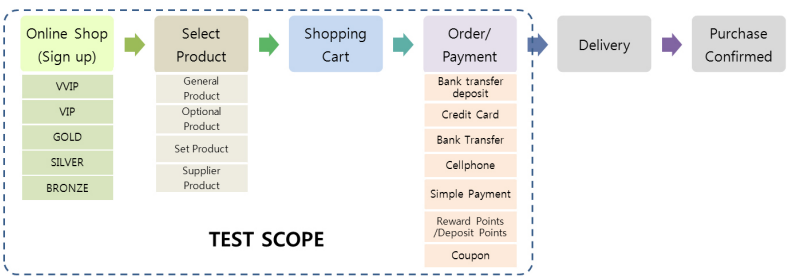
3.1 전자상거래 온라인 쇼핑몰 구매 과정

<Figure 3>은 전자거래 온라인 쇼핑몰 구매과정을 도식화 한 것이다. 일반적인 온라인 쇼핑몰의 구매과정은 고객이 쇼핑몰에 접속하여 로그인을 하고, 구매를 원하는 상품을 선택하여 장바구니에 담은 후에 주문서작성페이지로 이동하여 주문자정보와 배송정보를 입력하고 지불 수단을 선택 후 결제하면 주문이 완료된다. 판매자는 주문한 상품을 구매자에게 발송하고, 구매자가 배송된 상품을 받으면 구매확정 후 평가를 입력하면 모든 구매과정은 종료된다. 그런데, 구매자는 여러 등급의 회원으로 분류되고, 다양한 종류의 상품이 존재한다. 또한, 지불결제수단은 무통장입금, 카드결제, 계좌이체, 휴대폰결제, 간편결제, 예치금/적립금결제, 쿠폰결제, 간편결제 등 수많은 종류의 결제 수단이 사용된다.

본 연구는 모든 구매과정을 검증하는 것이 아니라 배송과 구매확정 및 평가 영역을 제외한 쇼핑몰 접속부터 주문서작성 페이지에서 지불결제수단을 선택하는 단계까지를 자동화 구현의 범위로 정하였다.

3.2 쇼핑몰 설정 구성 조합

테스트 자동화 실행을 위해 테스트 자동화 전용 쇼핑몰 구성이 필요하다. 생성한 쇼핑몰에는 상품 설정 조합, 회원 설정 조합, 배송비 설정 조합, 쿠폰할인 설정 조합, 적립금 설정 조합, 고객혜택 설정 조합 등 다양한 설정 조건들을 각 쇼핑몰들의 특성에 맞게 설정했다.



〈Figure 3〉 Process of E-Commerce

쇼핑몰의 모든 설정 조건의 조합이 불가능하여 운영환경에서 사용 중인 설정을 조사하고, 사용빈도가 높은 설정 항목들을 도출하여 등가분할, 경계값 분석, Pairwise(2-way) 기법을 적용하여 조건 조합을 생성하였다(Patton, 2006).

Pairwise 기법은 'Allpairs' 툴을 사용하였으며, 각 조합 조건 설정 SET에 대한 상세 내용은 아래와 같다.

1. 상품 설정 조합은 상품 공급 조건, 상품 세금 조건, 상품 옵션 조건, 상품 배송비 조건, 상품 조건, 상품 무이자 조건 등의 설정을 조합하여 261개의 상품을 생성하였다.
2. 회원 설정 조합은 회원등급 결제 조건, 상품별 할인과 중복 시 적용 조건, 회원등급 할인/적립 조건 등의 설정을 조합하여 52개의 회원등급을 생성하였다.
3. 배송비 설정 조합은 국내/해외 배송비 설정 조건, 무료배송비 우선 설정 조건, 상품별 개별 배송비 설정 조건, 공급사 배송비 설정 조건, 해외 배송 보험료 설정 조건, 지역별 배송비 설정 조건 등의 설정을 조합하여 59개의 배송비 조건을 설정하였다.
4. 쿠폰할인 설정 조합은 쿠폰 사용 조건, 쿠폰 사용 제한 조건, 적립금 동시 사용 조건, 할인 동시 사용 조건, 상품/주문서 동시 사용 조건, 쿠폰 종류 조건, 혜택 구분 조건, 발급 구분 조건, 사용범위 조건, 사용가능 결제수단 조건, 선택 범주 적용 조건 등의 설정을 조합하여 16가지의 쿠폰 사용 설정과 60개의 쿠폰을 생성하였다.
5. 적립금 설정 조합은 적립금 지급 시점 설정 조건, 적립금 절사 조건, 회원등급 추가 적립금 지급 설정 조건, 적립금으로 구매 시 적립 기준 설정 조건, 1회 사용 적립금 최대 사용한도 설정 조건 등의 설정을 조합하여 59개의 적립금 조건을 설정하였다.
6. 고객혜택 설정 조합은 고객혜택 유형 조건, 기간 유형 조건, 상품 범위 조건, 참여 대상 조건, 사용 범위 조건, 구매 횟수 조건, 구매 수량 조건, 제공 혜택 조건 등을 조합하여 206개의 고객혜택 조건을 설정하였다.

3.3 테스트케이스 시나리오 구성

<Figure 4>는 온라인 쇼핑몰에서 상품을 구매하는 하나의 시나리오를 엑셀 문서를 이용하여 테스트케이스로 생성한 예제이다. 아래의 테스트케이스 시나리오의 구성은 다음과 같다.

Code	Product		Mall Set	Coupon	Membership	Payment	Order
	P1	P2					
SM_0001	S_001		M_1	C_001	H_001	PC Card	D_01
SM_0002	S_002		M_1	C_002	H_004	PC Card	D_02
SM_0003	S_003		M_1	C_003	H_005	PC Bank Transfer	D_03
SM_0004	S_004		M_1	C_004	H_006	PC Bank Transfer	D_04
SM_0005	S_005		M_1	C_005	H_007	PC Card	D_05
SM_0006	S_006		M_1	C_006	H_008	PC Card	D_06

〈Figure 4〉 Testcase Scenario

1. Code : 테스트케이스를 구분하는 코드로 테스트 자동화 실행 시 테스트케이스를 맵핑하는 Key로 사용된다.
2. Product : 고객이 구매하고자 하는 상품을 지정하는 것으로, 상품 설정 조합에 의해 생성된 상품을 1개 또는 최대 2개까지 구매 가능하다.
3. Mall Set : 테스트케이스가 실행되는 몰을 의미한다.
4. Coupon : 구매 시 사용할 쿠폰으로 쿠폰할인 설정 조합에 의해 생성된 쿠폰을 적용한다.
5. Membership : 로그인 회원이 가진 회원등급으로 회원등급 설정 조합에 의해 생성된 회원등급은 회원등급 할인 혜택이 적용되며, 로그인 계정과 1:1로 맵핑된다.
6. Payment : 지불 결제 수단으로 PC와 모바일로 구분되며, 무통장, 카드, 실시간계좌이체, 적립금 전액결제, 휴대폰 결제, 네이버페이, 카카오페이, 페이코 등으로 구분된다.

7. Order : 결제 시 예치금, 적립금, 마일리지 등의 사용 여부를 코드화 하였다.

3.4 테스트케이스의 기대결과

테스트케이스의 기대결과는 장바구니 페이지, 주문서 작성 페이지, 주문완료 페이지가 있다. 각 페이지의 특정 영역을 실제 테스트 결과와 비교하여 검증한다.

Expected Result				
Cart				
Price	Shipping	Reward points 1	Reward points 2	Total Amount
11,000	0	1,300		11,000
10,000	0	1,300		10,000
10,000	0	1,300		10,000
10,000	0	1,300		10,000
11,000	0	1,300		11,000
10,000	0	1,300		10,000

〈Figure 5〉 Expected Result of Cart

장바구니 페이지에서는 <Figure 5>와 같이 상품합계금액, 배송비, 적립금, 총 주문합계 금액을 검증한다.

<Figure 6>은 테스트 자동화가 수행될 때 쇼핑몰의 장바구니 페이지에 노출된 정보를 기대결과와 비교하는 항목들을 표기한 것이다. 주문서 작성 페이지에는 가장 많은 검증 항목이 존재한다.

〈Figure 6〉 Cart Page

<Figure 7>은 상품합계 금액, 상품 할인 금액, 회원등급 할인 금액, 회원등급 적립금 금액, 배송비, 상품 적립금, 쿠폰 할인 금액, 쿠폰 적립금, 예치금, 적립금, 마일리지, 총 결제 금액, 결제 창 금액을 검증한다.

<Figure 8>은 테스트 자동화가 수행될 때 쇼핑몰의 주문서 작성 페이지에 노출된 정보를 기대결과와 비교하는 항목들을 표기한 것이다.

〈Figure 8〉 Order Page

Expected Result														
Order														
Place Order								Make Payment					Total Amount	Payment Amount
Price	Product Discounts	Membership Discounts	Membership Reward Points	Shipping	Reward Points 1	Reward Points 2	Total Amount	Coupon Discounts	Coupon Reward Points	Deposit	Reward Points	Mileage		
11,000	0	0	0	0	1,300		11,000	500	0	0	0	0	10,500	10,500
10,000	0	500	0	0	1,300		9,500	500	0	0	500	0	8,500	8,500
10,000	0	0	0	0	1,300		10,000	500	0	500	0	0	9,000	9,000
10,000	0	500	0	0	1,300		10,000	500	0	500	500	0	8,500	8,500
11,000	0	440	0	0	1,300		10,560	330	0	0	0	0	10,230	10,230
10,000	0	0	0	0	1,300		10,000	300	0	0	500	0	9,200	9,200

〈Figure 7〉 Expected Result of Order for Verification

Expected Result											
Order Complete											
Price	Reward Points 1	Reward Points 2	Product Discounts	Membership Discounts	Membership Reward Points	Coupon Discounts	Coupon Reward Points	Shipping	Use Deposit	Use Reword Points	Total Amount
11,000	1,300	0	0	0	0	500	0	0	0	0	10,500
10,000	1,300	0	0	500	0	500	0	0	0	500	8,500
10,000	1,300	0	0	0	0	500	0	0	500	0	9,000
10,000	1,300	0	0	500	0	500	0	0	500	500	8,500
11,000	1,300	0	0	440	0	330	0	0	0	0	10,230
10,000	1,300	0	0	0	0	300	0	0	0	500	9,200

〈Figure 9〉 Expected Result of Order Complete

주문완료 페이지에서는 <Figure 9>와 같이 상품 합계 금액, 상품 적립금, 상품 할인 금액, 회원등급 할인 금액, 회원등급 적립금, 쿠폰 할인 금액, 쿠폰 적립금, 배송비, 사용한 예치금, 사용한 적립금, 최종 결제 금액을 검증한다.

<Figure 10>은 테스트 자동화가 수행될 때 쇼핑몰의 주문완료 페이지에 노출된 정보를 기대결과와 비교하는 항목들을 표기한 것이다.

〈Figure 10〉 Order Complete Page

이와 같이 영역별 주요 페이지에 대한 기대결과를 엑셀로 정리하여 하나의 주문 시나리오를 테스트 케이스로 생성하였다. 그리고 해당 조건의 데이터를 이용하여 작성된 자동화 스크립트가 실행되면서 테스트 케이스의 기대결과와 실제 화면에 노출된 항목들을 비교하여 테스트 결과를 검증한다. 따라서 다양한 테스트 데이터의 조합 조건과 테스트 시나리오에 맞는 최소한의 자동화 스크립트만 있다면, 무수히 많은 조합의 테스트케이스를 생성하여 반복적으로 실행 할 수 있다.

3.5 테스트 자동화 실행 환경

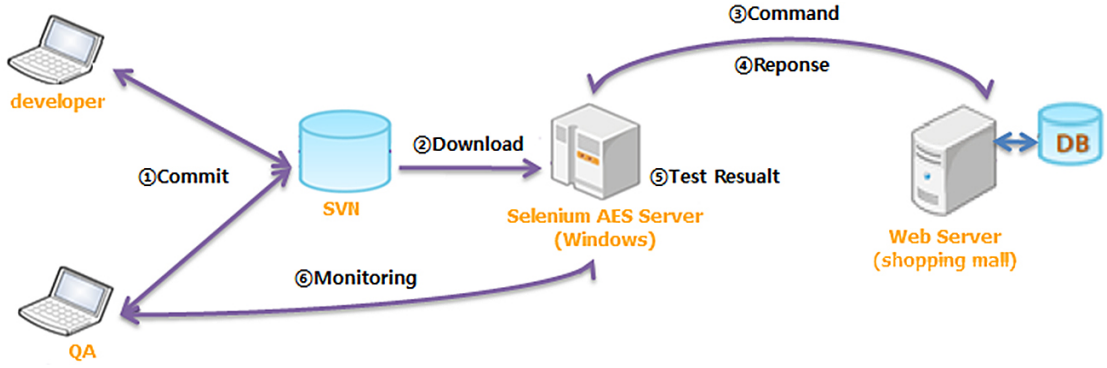
테스트 자동화 실행을 위한 기본적인 툴은 Selenium을 이용하였고, 개발자는 최소한의 쇼핑몰 구매 과정을 자동화 스크립트로 작성하였다. QA(Quality Assurance)는 다양한 설정 조합 조건을 기반으로 테스트 케이스를 생성하였고, 작성된 자동화 스크립트와 테스트 케이스는 SVN에 추가하고 관리한다.

Selenium AES 서버는 개발자가 작성한 자동화 스크립트와 QA가 작성한 테스트 케이스를 다운받아 작성된 모든 테스트 케이스의 데이터 정보를 기준으로 사전에 정의된 쇼핑몰에서 자동화 스크립트를 실행한다.

실행된 결과는 Selenium AES 서버에 전달되고 QA는 그 실행 결과를 확인한다.

QA는 오류가 발생하는 원인을 분석하여 실제 오류 인지, 자동화 스크립트 오류인지, 엑셀 데이터 오류인지를 판단하고 이슈의 원인에 따라 대응한다.

이와 같은 일련의 절차를 도시화 하면 <Figure 11>과 같다.



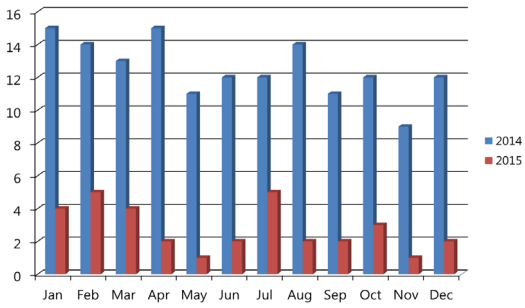
〈Figure 11〉 Running the Automated Tests

4. 실험 및 결과

본 연구에서는 온라인 쇼핑몰 결제수단의 효과적적인 회귀테스트를 위해 테스트 자동화가 구현되기 전인 2014년 데이터와 테스트 자동화가 적용된 2015년 데이터의 비교를 통해 통합테스트 환경에서의 결함 발견 활동과 운영환경의 결함 예방 활동에 대한 테스트 자동화 효과를 분석하였다.

<Figure 12>는 2014년 운영환경 결함 발생 건수와 2015년도 운영환경 결함 발생 건수를 비교한 그래프이다. 2014년 평균 결제 결함 발생 건수는 12.5건이고, 2015년 평균 결제 결함 발생 건수는 2.75건으로 약 1/4로 결함이 감소했음을 알 수 있다.

<Figure 13>은 2014년과 2015년 운영환경에서 발생한 결함 건수를 독립표본 검증을 통해 통계적으로 유의미한 차이가 있는지 분석한 결과이다. 먼저 Levene의 등분산 검정 결과, 유의확률 .551로 .05보다 크므로 등분산이 인정된다. 따라서 등분산이 유지된 경우의 유의확률 .000은 .05보다 작으므로 2014년과 2015년은 통계적으로 차이가 있고, 운영환경에서 결함 발생률이 줄어들었음을 나타낸다.

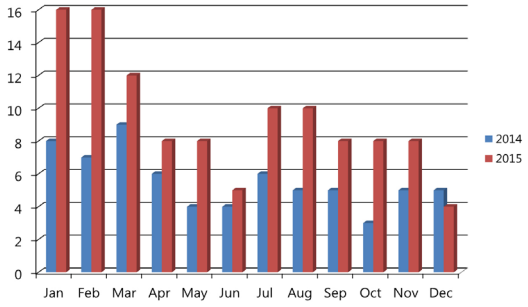


〈Figure 12〉 Comparison of Live Defects(2014 vs 2015)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means				
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std.Error Difference
Defect_Count	Equal variances assumed	.366	.551	14.805	22	.000	9.750	.659
	Equal variances not assumed			14.805	20.960	.000	9.750	.659

〈Figure 13〉 T-Test of Live Defects



〈Figure 14〉 Comparison of Test Defects

<Figure 14>는 2014년 통합 테스트 환경에서 발견한 결함 건수와 2015년 통합 테스트 환경에서 발견한 결함 건수를 비교한 그래프이다. 2014년 통합 테스트에서 발견한 결함의 평균은 5.6건 2015년 통합 테스트에서 발견한 결함의 평균은 9.4로 약 2배 정도 많은 결함을 검출하였다.

<Figure 15>는 2014년과 2015년 통합 테스트 환경에서 발견한 결함 건수를 독립표본 검증을 통해 통계적으로 유의미한 차이가 있는지 분석한 결과이다.

Levene의 등분산 검정 결과는 등분산이 인정되므로, 결함 건수가 같을 유의확률을 확인하면 0.004로 .05보다 작으므로 2014년과 2015년은 통계적으로 차이가 있음을 알 수 있다. 2014년보다 2015년에 통합테스트 환경에서 더 많은 결함을 발견하였으므로 테스트 자동화를 적용한 후 통합 테스트 결함 검출률이 높아졌기 때문에 운영 환경에서는 결함 발생률이 줄어들었다.

이와 같이, Record-Play Back 기법과 데이터 주도 접근법을 활용한 회귀테스트 자동화 적용으로 기존 수동 테스트로는 수행할 수 없었던 수많은 테스트 케이스를 반복적으로 실행하고, 전체적인 테스트 수행 시간을 단축하므로 빠른 결과 확인과 피드백이 가능해졌다. 또한, 객관적인 테스트 평가 기준으로 테스트 결과에 대한 신뢰성을 제공하고, 여러 영역의 잠재된 결함들을 빠르게 발견하여 수정하므로 서비스의 품질을 향상시켰다. 이를 통해 회귀테스트 자동화 구현의 효과성과 효율성을 확인하였고, 이와 같은 방법을 응용하면 온라인 쇼핑몰처럼 특정 시나리오 안에서 다양한 데이터의 조합 조건에 따라 테스트가 필요한 영화 또는 공연 예매 사이트, 항공권 예약 사이트, 인터넷 검색 사이트, 인터넷 게시판 등 다양한 서비스 영역에서 구현된 방법과 유사하게 적용이 가능할 것으로 예상되며, 이를 통해 본 실험 결과와 같이 기존보다 확대된 테스트 범위와 빠른 테스트 수행은 서비스 시스템의 품질향상에 도움을 줄 수 있을 것으로 기대한다.

5. 결론 및 향후 연구 과제

온라인 쇼핑몰은 전자 상거래의 급속한 성장과 웹 기술의 발달로 다양한 지불 결제수단을 제공하며 점점 더 복잡해지고 있다. 또한, 다양한 결제 방식의 지원으로 결제수단에 대한 품질 및 신뢰성

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means				
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std.Error Difference
Defect_Count	Equal variances assumed	4.167	.05.	-3.231	22	.004	-3.833	1.187
	Equal variances not assumed			-3.231	15.526	.005	-3.833	1.187

〈Figure 15〉 T-Test of Test Defects

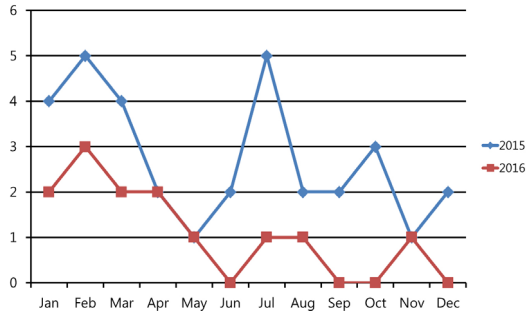
의 요구가 증가하고 있다. 이러한 요구사항을 충족시키기 위해 웹 어플리케이션 개발 과정에서 강도 높은 테스트가 필요하며, 소프트웨어의 변경에 따른 신규 기능 추가 및 기존 기능 수정으로 새롭게 유입되는 결함을 해결하기 위한 회귀 테스트는 반드시 수행되어야 한다. 하지만 회귀 테스트는 테스트 범위가 넓고, 반복적으로 테스트해야하기 때문에 실무에서는 회귀 테스트가 제대로 수행되지 않거나, 선택적으로 적은 테스트 케이스만 수행하는 경우가 대부분이다.

본 연구에서는 Record-Play Back 기법의 GUI 자동화 도구인 Selenium 툴과 데이터 주도 접근 방법을 이용한 온라인 쇼핑몰 테스트 자동화 방법을 제안하고, 온라인 쇼핑몰 구매 결제 시스템에 이를 적용하여 효과적으로 회귀 테스트를 수행하였다.

제안된 회귀테스트 자동화 구현 방법은 3단계로 이루어진다. 첫 번째 단계는 Selenium 툴을 통해 온라인 쇼핑몰 구매 과정을 스크립트로 구현한다. 두 번째 단계는 데이터 주도 접근법을 통해 테스트 실행에 필요한 모든 데이터 정보를 엑셀 파일로 정의하여 테스트케이스를 생성한다. 마지막으로 Selenium AES를 이용하여 다양한 브라우저 환경에서 생성한 모든 테스트케이스를 테스트하고, 테스트 수행 결과를 QA가 확인한다.

테스트 자동화가 구현되기 전 2014년에는 결제수단 테스트를 카드, 무통장, 계좌이체, 휴대폰 결제에 대해 각각 10가지 조건으로 40개 정도의 테스트케이스를 수행했지만, 2015년 자동화 초기에만 약 1,000건 정도의 테스트케이스를 추가해 결제수단 테스트를 진행했다. 운영환경 이슈에 대한 테스트 케이스 추가 및 신규 결제수단 추가, 다양한 솔루션 설정 조건들의 변경으로 현재는 약 15,646건의 결제수단 테스트케이스를 통합테스트 기간 동안 매일 실행하고 있다.

<Figure 16>은 2015년과 2016년에 운영환경에서 발생한 결함 건수를 비교한 그래프이다. 이처럼 아직까지도 운영 환경에서 결함이 발생하는 이유는



〈Figure 16〉 Comparison of Live Defects(2015 vs 2016)

작성된 테스트케이스가 결제수단 전체 범위를 포함하지 못하기 때문이다. 현재의 조합 조건인 상품 설정 조합 261가지, 회원 설정 조합 52가지, 배송비 설정 조합 59가지, 쿠폰 할인 설정 조합 60가지, 적립금 설정 조합 59가지, 고객혜택관리 설정 조합 206가지를 모두 곱하면 583,937,471,520가지의 테스트케이스가 생성되어야 한다. 하지만 이 모든 테스트케이스를 생성하는 것도 어려울 뿐만 아니라 테스트 실행을 위해서도 엄청난 시간과 비용이 투자되어야 하기에, Pairwise(2-way) 기법을 적용하여 테스트케이스를 생성하였으며, 지속적으로 테스트케이스를 늘려나갈 예정이다.

본 연구는 다음과 같은 실무적 공헌과 한계점을 제시할 수 있다. 먼저 실무적 공헌은 첫째, Record-Play Back 기법의 테스트 자동화는 유지보수 비용이 매우 높다는 문제점을 가지고 있다. 하지만 테스트 실행 코드와 데이터의 분리로 테스트 케이스 증가 여부에 관계없이 결제 시나리오만 자동화 스크립트로 작성하기 때문에 유지보수 비용을 감소시켰다. 둘째, 테스트 커버리지 확대를 위한 테스트 케이스 추가 작업이 용이하다. 이는 기존 작성된 자동화 스크립트의 추가 변경 없이 엑셀 문서의 데이터 조합 조건을 통해 쉽게 테스트 케이스를 추가, 수정, 삭제할 수 있는 장점을 가지고 있다. 셋째, 수동 테스트에서 발생하는 테스터의 실수나 누락이 발생하지 않고, 테스트 수행 결과에 대해 빠른 피드백을 제공하므로 개발 시간을 단축하는데 도움을 줄 수 있다. 또한, 테스트 자동

화 수행 결과는 객관적인 평가 기준을 제시하고, 테스트 결과에 대한 신뢰성을 제공할 수 있다.

한계점은 첫째, 복잡한 테스트 케이스를 테스트 엔지니어가 직접 수동으로 작성하는데 많은 시간과 노력이 소요된다. 특히 데이터의 조합 조건의 항목들을 도출하고, 정의하여 기대결과를 작성하는 것은 고도의 지적 노력을 요하는 힘든 작업이다. 둘째, 자동화 스크립트 작성과 테스트 자동화 환경을 구축하는 것은 개발자의 도움을 받아야 한다는 것이다. 테스트 엔지니어는 개발자와 같은 정교한 자동화 스크립트 작성이 불가능하고, 테스트 자동화 환경에 대한 이해도 부족하기 때문이다. 셋째, 자동화 실행 결과 실패가 발생한 항목에 대한 원인이 실제 결함인지, 자동화 스크립트 오류인지, 테스트 케이스의 기대결과 오류인지, 테스트 실행 환경 문제인지에 대한 분석이 필요하다. 이를 위해 오류가 발생하는 부분에 대한 녹화와 이미지 캡처 기능을 추가했지만, 기본적으로 자동화 스크립트 코드를 기반으로 분석하는데 어려움이 존재한다.

이러한 한계점을 해결하기 위해서는 데이터 조합 조건을 이용해 자동으로 효과적인 테스트 케이스를 도출하는 방법과 자동화 실행 결과 노출되는 실패 항목에 대한 효율적인 분석 방법을 위한 연구가 추가적으로 필요하다.

References

Bolt, W., D. Humphrey, and R. Uittenbogaard, “Transaction Pricing and the Adoption of Electronic Payments : A Cross-Country Comparison”, *International Journal of Central Banking,* Vol.4, No.1, 2008, 89-123.

Chae, H.S., “Model-based Test-Concepts and Issues”, *Communications of the Korean Institute of Information Scientists and Engineers,* Vol.32, No.4, 2014, 59-71.
(채흥석, “모델 기반 테스트-개념과 이슈”, *정보과학회지,* 제32권, 제4호, 2014, 59-71.)

Damm, L.O. and L. Lundberg, “Results from Introducing Component-level Test Automation and Test-Driven Development”, *Journal of Systems and Software*, Vol.79, No.7, 2006, 1001-1014.

DMC Media, “2016 The Market Trends of Mobile Easy Payment Services”, DMC, 2016.
(DMC Media, “2016 모바일 간편결제 서비스 시장 현황과 전망”, DMC, 2016.)

Do, H. and G. Rothermel, “On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques”, *IEEE Transactions On Software Engineering*, Vol.32, No.9, 2006, 733-752.

Elbaum, S., A.G. Malishevsky, and G. Rothermel, “Prioritizing Test Cases for Regression Testing”, *Proceedings of the International Symposium of Software Testing and Analysis,* 2000, 102-112.

Gerrard, P., “Testing GUI Applications”, *The Fifth EuroSTAR(Software Testing and Review) Conference,* 1997, 24-28.

Hwang, Y. and J. Jeong, “Electronic Commerce and Online Consumer Behavior Research : A Literature Review”, *Information Development,* Vol.32, No.3, 2016, 377-388.

Jeong, K.C., S.H. Lee, J.H. Jo, Y.M. Sin, and Y.O. Park, “An Efficient Regression Testing Using Testing Automation Tool for Industrial Automation Software”, *Proceedings of Control Automation System Symposium*, 2008, 245-249.
(정광철, 이상훈, 조주현, 신영민, 박용운, “자동화 테스트 도구를 활용한 산업용 자동화 소프트웨어의 효율적인 회귀 테스트”, *제어로봇시스템학회 국내학술대회논문집,* 2008, 245-249.)

Jin, J.S., H.M. Kim, and J.S. Park, “A Study on Behavior in Using Fin-Tech Based on Life

Style Types", *Journal of Information Technology Services,* Vol.16, No.1, 2017, 119-138.
(진정숙, 김현모, 박주석, "라이프스타일 유형에 따른 모바일 간편결제 서비스의 이용행태 연구", *한국IT서비스학회지*, 제16권, 제1호, 2017, 119-138.)

Jo, H. and J.M. Lee, "A Study on Antecedents of WOM in the Context of Internet E-Commerce", *Journal of Information Technology Services,* Vol.12, No.2, 2013, 231-242.
(조 현, 이정민, "인터넷 전자상거래 환경에서의 구전효과의 선행 요인에 관한 연구", *한국IT서비스학회지,* 제12권, 제2호, 2013, 231-242.)

Jung, W.J., S.R. Rah, and Y.L. Choi, "A Study on the Selection of Test Scope and the Prioritization of Test Case Based on Modification Method for Regression Testing", *Journal of Information Technology Services,* Vol.14, No.2, 2015, 129-142.
(정우진, 나상린, 최용락, "변경 메서드 기반의 회귀 테스트 검증 범위 선택 및 검증 항목 우선순위 선정에 관한 연구", *한국IT서비스학회지*, 제14권, 제2호, 2015, 129-142.)

Kim, S.D., P. Park, and S.B. Yang, "Influencing Factors on Users' Resistance to the Mobile Easy Payment Services : Focusing on the Case of KakaoPay Users", *Journal of Information Technology Services*, Vol.16, No.2, 2017, 139-156.
(김소담, Philip Park, 양성병, "모바일 간편결제 서비스에 대한 사용자 수용저항 요인 : 카카오페이 사용자를 중심으로", *한국IT서비스학회지*, 제16권, 제2호, 2017, 139-156.)

Kwon, W.I., H.J. Lee, S.H. Choi, S.H. Lee, and E.Y. Park, *Practical Software Testings for the Developers,* Third Edition, STA, 2010.
(권원일, 이현주, 최승희, 이승호, 박은영, *개발자도 알아야 할 소프트웨어 테스팅 실무*, 3판, STA, 2010.)

Lee, J.G., H.S. Kim, S.H. Kuk, and D.W, Cho, "Record-Playback based Automatic Test Case Generation for GUI Test", Proc. of the *KIISE Korea Computer Congress*, 2007, Vol.34, No.1(B), 2007, 96-100.
(이정규, 김현수, 국승학, 조대환, "Record-Playback 기술 기반의 GUI 테스트 케이스 자동 생성", *한국컴퓨터종합학술대회논문집,* 제34권, 제1호(B), 2007, 96-100.

Memon, A., A. Nagarajan, and Q. Xie, "Automating Regression Testing for Evolving GUI Software", *Journal of Software Maintenance and Evolution : Research and Practice,* Vol.17, No.1, 2005, 27-64.

Meszaros, G., R. Bohnet, and J. Andrea, "Agile Regression Testing Using Record and Playback" In F. Maurer & D. Wells(Eds.), Extreme Programming and Agile Methods-XP/Agile Universe, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Vol.2753, 2003, 111-119.

Morris, S., "Automated GUI Testing", Tessella Support Services PLC, 1999.

Patton, R., *Software Testing,* Second Edition, Information publishing Group, 2006.
(Ron Patton, *소프트웨어 테스팅,* 제2판, 김도균 옮김, 정보문화사, 2006.)

Polo, M., S. Tendero, and M. Piattini, "Integrating Techniques and Tools for Testing Automation", *Software Testing Verification and Reliability,* Vol.17, No.1, 2007, 3-39.

Prototech Solutions, "9 Tips for selecting test cases for Regression Testing", available at http://www.prototechsolutions.com/regression-testing/(Accessed on May 18, 2017).

Rothermel, G., S. Elbaum, A. Malishevsky, P.

Kallakuri, and X. Qiu, “On Test Suite Composition and Cost-Effective Regression Testing”, *ACM Transactions on Software Engineering and Methodology*, Vol.13, No.3, 2004, 277-331.

Schwartz, A. and H. Do, “Cost-effective Regression Testing through Adaptive Test Prioritization Strategies”, *Journal of Systems and Software,* Vol.115, 2016, 61-81.

w3ii.com, “Data Driven Testing”, available at http://www.w3ii.com/software_testing_dictionary/data_driven_testing.html (Accessed on May 15, 2017).

Zech, P., P. Kalb, M. Felderer, C. Atkinson, and R. Breu, “Model-based Regression Testing by OCL”, *International Journal on Software Tools for Technology Transfer*, Vol. 19, No.1, 2017, 115-131.

About the Authors

Sungyong Kim (loveisksy@naver.com)

Sungyong Kim is currently a Ph.D. student at the Department of Digital Management, Korea University and the quality assurance engineer at Nexon Korea. He has received master's degree in Department Of Computer Information and Communication from Kangwon National University in 2013. His current research interests include automation testing, regression testing, and software engineering.

Daihwan Min (mismdh@korea.ac.kr)

Daihwan Min is a professor of Digital Management at Korea University Sejong Campus in Korea. He has a BBA degree from Seoul National University, a Master degree in industrial engineering from KAIST, and a Ph.D. degree in business administration from the University of Michigan. His research interests include diverse areas such as business process management, information privacy, digital management, systems analysis, service management, and business ecosystem.

Seongtaek Rim (misrim@korea.ac.kr)

Seongtaek Rim is a professor of Digital Management at Korea University, Sejong Campus. He received his M.A. and Ph. D degrees in Computer Information Systems from Georgia State University. His research interests span the business value of information technology investment; telecommunication business models and strategy; national ICT policies; mobile business, and so on.