韓國IT서비스學會誌
第14卷 第2號
2015年 6月, pp.129-142

Journal of Information Technology Services
http://dx.doi.org/10.9716/KITS.2015.14.2.129

변경 메서드 기반의 회귀 테스트 검증 범위 선택 및 검증 항목 우선순위 선정에 관한 연구*

정우진** · 나상린*** · 최용락****

A Study on the Selection of Test Scope and the Prioritization of Test Case Based on Modification Method for Regression Testing*

Woo-Jin Jung** · Sang-Rin Rah*** · Yong-Lak Choi****

■ Abstract ■

The purpose of this study is to suggest an effective regression testing method in order to minimize the scope of test resulting from the modification of software and to prevent mismatch of test case and test objects.

As a way to improve the efficiency of regression testing which uses a change-centric testing technique, the method flow is analyzed and grasped through a static analysis based on source code in order to identify modified parts. After the order of priority is set according to the results of user action log-based dynamic analysis on identified regression testing objects, test effect can be raised by adjusting the order of priority using code complexity.

Quality assurance coverage can be checked using the user action log suggested in this study, and the progress of test and whether or not each function has been verified can be checked, too. In addition, by minimizing test parts and adjusting the order of test, costs and time can be saved, making it possible to conduct regression testing effectively.

Keyword : Regression Testing, Test Scope Selection, Test Case Prioritization, Test Coverage

Submitted : January 28, 2015 1st Revision : June 1, 2015 Accepted : June 10, 2015

* 본 논문은 모든 저자가 같게 공헌하였음.
** 숭실대학교 정보과학대학원, 주저자
*** 숭실대학교 SW특성화대학원, 공동저자
**** 숭실대학교 SW특성화대학원, 교신저자

1. 서 론

소프트웨어 테스트는 소프트웨어의 문제점을 발견하고 수정해서 제품의 품질을 개선하기위한 활동이다. 소프트웨어 시스템은 규모 및 복잡성이 증가하고 있다. 그렇기 때문에 전체를 검증하기에는 현실적으로 많은 시간과 비용을 필요로 한다. 이와 같은 한계를 극복하기 위해서는 일단 오류가 발생할 가능성이 높은 테스트 범위를 선정한다. 그리고 검증 항목에 대한 개수를 최대한 적게 도출해 내는 방법이 필요하다(Leung and White, 1989).

소프트웨어는 배포 이후 사용자의 요청, 오류 수정, 성능 개선 등으로 인해 끊임없이 변경된다. 이러한 변경이 발생할 때마다 품질 저하를 막기 위한 회귀 테스트(Wikipedia, 2014)는 반드시 수행되어야 한다. 회귀 테스트는 소프트웨어 유지 보수 측면에서 많은 비용과 시간을 요구하지만, 소프트웨어 안정성을 위해 필수적으로 수행해야 하는 활동이다. 그러나 소프트웨어의 수정된 부분의 규모와는 상관없이 전체 범위를 반복해서 테스트 한다는 것은 현실적으로 어렵다. 이 때문에 모든 검증 항목을 확인하는 전체 회귀 테스트를 수행하지 않고, 변경된 항목과 영향 받는 부분에 대해서만 요구사항 명세서를 확인하고 재검증한다(Rothermel et al., 2001).

본 논문은 회귀 테스트의 효율성 향상에 연구 중점을 두고 있다. 변경코드 기반의 메서드 흐름을 분석하여 수정된 프로그램에 따라 검증 범위와 검증 항목을 도출하고, 사용자의 활동에서 누적된 결과와 코드 복잡도를 기반으로 우선순위를 부여한다. 그러면 테스트 항목을 가감할 수 있어 사용자가 소프트웨어의 오류를 발견할 확률을 줄일 수 있다(Mei, et al., 2009). 이를 기반으로 변경 중심의 효율적인 회귀 테스트 범위를 산정한다. 그리고 호출되지 않는 코드를 파악한 후, 해당 부분은 검증 대상에서 제외하거나 우선순위에 따라 최소화함으로써 테스트의 효율성을 향상시킬 수 있고 비용 대비 시간을 절약할 수 있다.

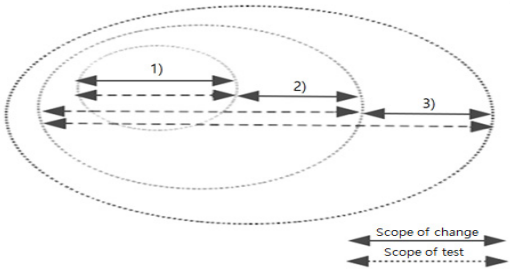
2. 관련 연구

2.1 회귀 테스트의 문제점

회귀 테스트는 소프트웨어 개발이 완료된 이후 프로그램에 오류가 있는지 반복적으로 점검하는 테스트다. 회귀 테스트가 반복적으로 필요한 이유는 결함이 정확히 수정되었더라도 시스템이 의도하지 않았던 오동작이나 새로운 형태의 결함이 발생할 수 있기 때문이다. 이는 소프트웨어 모듈의 수정 및 추가 과정에서 기존의 소프트웨어에 악영향을 끼쳐 결함이나 오류를 초래하는지를 재검증하기 위한 목적으로 수행된다.

회귀 테스트 하는 가장 기본적인 방법은 프로그램 변경 전후의 검증 항목을 모두 확인하는 것이다. 그러나 소프트웨어의 기능 및 요구사항이 추가될수록 수행해야 하는 회귀 테스트 검증 항목도 늘어나게 되어 비용과 시간이 증가한다는 단점이 있다.

<Figure 1>은 소프트웨어의 변경이나 추가가 발생함에 따라 회귀 테스트의 검증 범위가 전체 범위로 늘어나는 것을 보여주고 있다. 소프트웨어의 오류 또는 사용자의 변경 요청이 발생할 때 마다 검증 범위는 점차 증가하게 된다. 회귀 테스트의 검증 범위가 증가 하면 변경 범위는 <Figure 1>과 같이 1), 2), 3) 순으로 추가된다. 하지만 시간과 비용의 문제로 변경된 부분만을 검증하게 된다. 이러한 문제를 해결하기 위해 검증 범위를 축소하거나 검증 항목의 우선순위를 부여하여 적용한다.



〈Figure 1〉 Regression Testing Scope

소프트웨어의 변경으로 인해 회귀 테스트 검증 범위가 증가하는 문제와 함께 나타나는 또 다른 문제는, 검증 대상과 검증 항목의 불일치로 인해 시간과 비용을 낭비하게 된다는 것이다. 변경이 발생한 부분에 대한 검증 항목이 누락되거나 정확한 데이터로 검증되지 않는다면, 프로그램이 내포하고 있는 오류나 결함을 식별하여 수정하기 어렵다. 회귀 테스트는 반복적으로 수행해야 하기 때문에 검증 항목은 회귀 테스트에서 가장 중요한 요소이며, 변경된 소프트웨어와 항상 동일한 검증 항목을 유지해야 한다. 따라서 소프트웨어의 변경과 함께 검증 항목이 추가, 수정, 삭제되어야 하지만 누락될 가능성은 항상 잠재되어 있다.

2.2 정적 테스트 기법

소프트웨어 테스트 기법은 정적 테스트와 동적 테스트로 구분된다. 정적 테스트란 소프트웨어를 실행하지 않고 테스트를 수행하는 방법으로 정적 테스트 수행만으로 초기에 결함을 발견할 수 있어 결함, 수정, 비용을 줄일 수 있고 시간적인 면에서도 효과적이다. 정적 테스트 기법 중 메서드 흐름 분석은 정적 분석 도구를 이용하여 프로그램에서 실행되는 각 메서드가 호출되는 과정을 분석하는 것으로, 동적 테스트에서 찾기 힘든 결함을 발견할 수 있다. 정적 분석 도구에 의해 하나의 프로그램을 구성하고 있는 단위 프로그램들 사이의 호출 관계를 call graph[1]라고 한다.

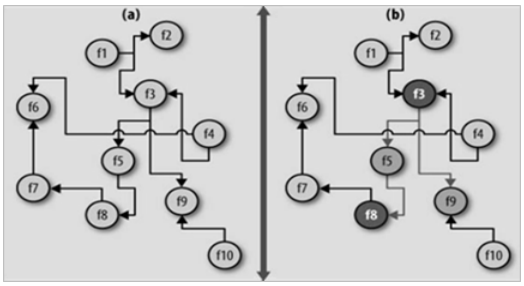
정적 테스트를 위해 분석된 메서드는 실제로 변경된 부분에 영향을 주는 메서드를 찾아내는 과정이 요구된다. 변경 영향성 파악을 위해서 메서드를 구분하면 <Table 1>과 같다(Lee and Kwon, 2011).

메서드의 호출 관계를 정확히 파악하고 검증 항목과 소스 코드와의 의존관계를 분석하면, 어떤 검증 항목이 어느 파일의 특정 메서드에 해당하는지 맵핑이 가능해진다(Tip and Jens, 2000).

1) call graph-호출 그래프.

〈Table 1〉 Method Classification for Call Relations

Method type	Description
Source function	◦ Called by other method, but no other dependencies exist ◦ Which means, no other call methods exist among the codes, except for System call, and Library call
Sync function	◦ Calls other methods(doesn't be called) ◦ Has no effect on other methods than itself
Pipe function	◦ Can either call methods, or be called ◦ Affected by the change in methods called



〈Figure 2〉 Regression Testing Scope Example

<Figure 2>에서 (a)는 변경 전 메서드의 호출관계를 표현한 예이고, (b)는 f3을 변경이 발생한 메서드로 가정하고 f3이 호출하는 메서드들과 f3을 호출하는 메서드가 다시 호출하는 메서드들이 영향성을 표현한 예이다(Tim and Adam, 2009). 회귀 테스트 시 검증 항목은 변경된 메서드와 영향 받는 메서드에 해당 하는 기능을 추출한다. 추가적인 검증 항목은 사용자 활동 로그에서 누적된 데이터를 기반으로 한다.

2.3 동적 테스트 기법

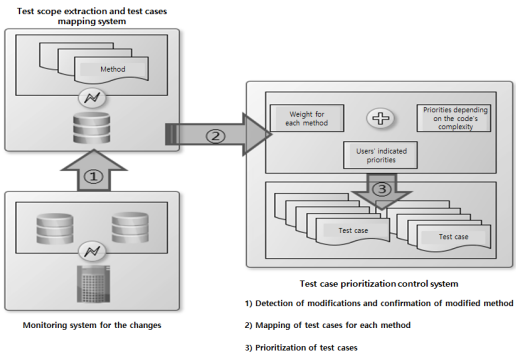
동적 테스트 기법이란 사용자가 소프트웨어를 직접 실행시켜 보면서 소프트웨어가 예상한 결과대로 작동하는지의 여부를 판단하는 테스트 방법으로 블랙박스 테스트와 화이트박스 테스트로 분류된다(Kim, 1999). 본 논문에서는 블랙박스 테스트에 속

한 랜덤(Random) 테스트 또는 애드-혹(Ad-Hoc) 테스트의 방법을 활용해 사용자 활동 로그를 적용한 동적 테스트를 수행하여 검증 항목 우선순위 지표로 활용한다.

사용자 활동 로그란 웹 사이트 또는 어플리케이션에서 사용자의 행위 패턴을 파악할 수 있는 로그 정보이다. 사용자 활동의 정의는 사용자가 화면상의 원하는 객체를 선택하거나 특정 항목을 실행하기 위해서 선택하는 일련의 흐름 정보들이다(Kim, 2004). 실제 사용자의 사용 행위를 기록한 로그 분석을 통해 사용자의 행동 패턴을 분석하고 호출된 메서드와 상호 대응하여 검증 항목의 가중치를 높일 수 있다.

3. 회귀 테스트 검증 항목 추출

회귀 테스트 시, 품질보증 담당자는 정적 분석을 통해 변경된 메서드와 영향 받는 메서드를 확인하여 검증 범위가 선정되면 변경할 검증 항목을 수정하고 누락된 검증 항목을 추가하는 작업을 한다(Wikipedia, 2014). 검증 범위와 검증 항목이 준비되면 우선순위가 부여된 검증 항목으로 회귀 테스트를 수행하게 된다. 테스트를 수행하는 동안 사용자 활동 로그를 수집하는 DB에는 지속적으로 로그를 남기게 되고 해당 로그를 이용하여 품질보증 테스트 커버리지 측정이 가능하다.

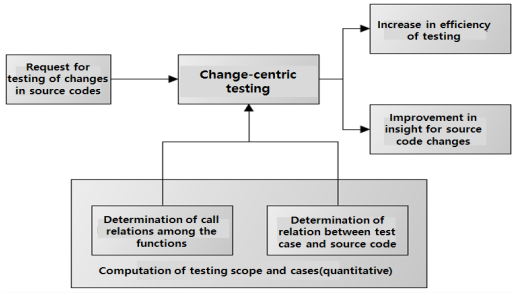


〈Figure 3〉 System Composition

<Figure 3>은 변경된 메서드와 영향 받는 메서드를 분석하여 가중치를 부여하고 사용자 로그와 코드 복잡도를 활용하여 우선순위를 부여한 검증 항목을 추출하는 회귀 테스트 시스템 구성도이다. 변경 사항 모니터링 시스템에서는 변경 파일을 체크하는 스크립트가 실시간으로 변경되는 파일을 모니터링 한다. 변경 코드가 감지되면 변경 메서드를 확인하여 검증 항목과 맵핑하고 메서드별 가중치와 사용자 활동 로그, 코드 복잡도를 분류하여 우선순위를 조정한다.

3.1 검증 범위 및 검증 항목 관리

본 연구에서는 회귀 테스트 시 효과적인 검증 범위 산정을 위해 변경이 발생한 부분과 그로 인해 영향 받는 부분을 메서드 간 호출 관계를 이용하여 소스 코드와 검증 항목의 의존관계를 파악함으로써 검증 효율성을 높인다(Seo, 2013). 요구사항 추가 또는 기능 변경으로 코드 수정이 발생하여 회귀 테스트 수행 시 검증 범위를 산정할 때, 변경-중심 테스팅 방법을 이용해서 검증했을 때의 장점을 도식화하면 <Figure 4>와 같다(Tim and Adam, 2009).

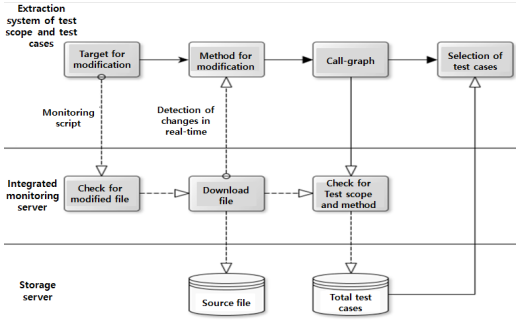


〈Figure 4〉 Change-Centric Testing

<Figure 4>와 같이 정량적인 검증 범위를 산정하기 위해, 품질보증 담당자는 메서드의 호출관계를 파악하고 검증 항목과 소스 코드의 관계를 분석하면 테스트의 효율성을 높일 수 있다.

<Figure 5>는 변경 사항을 감지하고 정적분석을

통한 call graph를 생성하여 변경된 메서드를 확인하고 검증 범위 및 검증 항목을 선정하는 과정을 도식화한 내용이다. 검증 범위의 메서드가 확인되면 전체 검증 항목에서 변경된 메서드에 해당하는 검증 항목을 추출하는 과정을 통해 검증 항목 선정을 한다.



〈Figure 5〉 Modifications Monitoring, Test Cases

메서드의 호출 관계가 파악되면 변경된 메서드와 변경된 메서드에 영향 받는 메서드를 구분하여 중요도를 결정하여 우선순위를 부여한다. 변경이 직접적으로 발생한 메서드는 회귀 테스트의 중요 검증 범위로 우선순위가 가장 높다. 1단계로 영향 받는 메서드도 결함을 내포할 가능성이 높고, 2단계로 영향 받는 메서드도 우선순위를 높여서 검증 대상에 포함해야 한다. 3단계 이상 영향 받는 메서드는 중복으로 처리될 수도 있고, 일반적으로 2단계까지만 문제가 없다면 그 이후로는 영향을 받지 않는 것으로 간주한다.

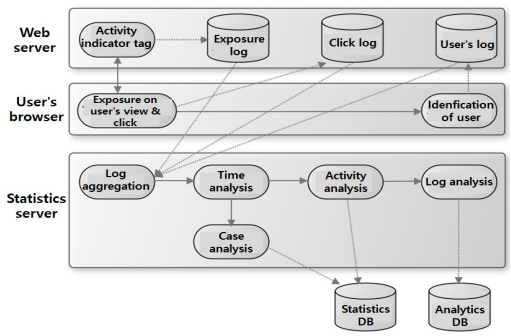
〈Table 2〉 Weights for Call Relations of Methods

Methods	Description	Effects	Weight
Changed method	Method changed by direct code modification	High	5
Stage1-affected method	Method which calls changed method	Average	3
Stage2-affected method	Upper method of a method calling changed method	Low	1

<Table 2>는 변경된 메서드의 호출관계별 평가 가중치의 영향도를 상, 중, 하로 분류하고 가중치를 '상'은 5, '중'은 3, '하'는 1로 정의하였다. 이에 근거하여 메서드 호출 관계별 가중치는 사용자 활동 로그 가중치와 합산하여 추출된 검증 항목의 우선순위 지표로 사용하고, 우선순위가 같은 메서드를 최소화하기 위해 최종적으로 코드 복잡도를 이용하여 우선순위를 조정한다.

3.2 검증 항목 우선순위 관리

사용자의 활동 로그를 검증 항목과 맵핑하여 활용하면 사용자가 주로 사용하는 패턴을 보이는 항목에 우선순위를 높일 수 있다.



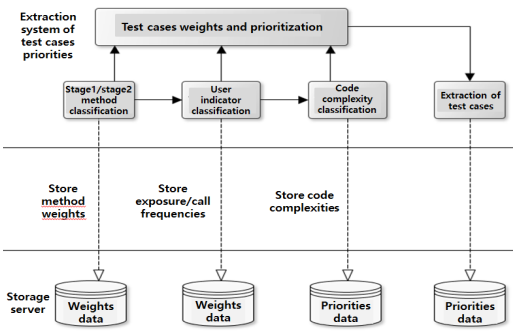
〈Figure 6〉 Log Analyzing System Composition

<Figure 6>과 같이 검증 항목의 우선순위를 적용하고 분석하기 위해서는 웹 서버, 통계 서버, 통계 DB, 분석 DB로 구성된 물리적인 시스템 영역이 필요하다. 사용자는 브라우저와 웹 서버 사이의 상호 작용에 의해 노출 로그, 클릭 로그, 사용자별 로그가 웹 서버에 남게 된다. 검증 항목의 우선순위를 사용자 행위 시나리오인 서비스 페이지별 노출 로그와 영역별 특정 활동 로그인 클릭 로그로 나누어서 적용한다. 검증 시간과 일정이 부족할 때 사용자 활동 로그 지표를 활용하여 우선순위를 조정해서 테스트를 수행하면, 서비스를 이용하는 사용자가 자주 사용하는 영역과 기능에 대해서는 선 검수가 이루어져 오류를 발견할 확률을

낮출 수 있다.

사용자 활동 로그를 분석한 후 변경된 코드와 영향 받는 코드에 해당하는 검증 항목은 실행 우선순위를 높게 조정한다. 메서드별 가중치를 부여한 뒤 추출된 검증 항목의 검증 우선순위는 누적된 사용자의 활동 로그를 기반으로 가중치를 1회 더 부여한다. 우선순위의 중복을 최소화하기 위해 최종적으로 코드 복잡도를 활용하여 우선순위를 세부적으로 조정한다(Thomas, 2008).

변경 또는 영향받는 메서드에 포함되지 않은 메서드에 대해서도 전체 회귀 테스트를 고려하여 호출 또는 노출빈도에 따라 가중치를 부여하여 사용한다. 전체 검정 범위에 대해 테스트가 수행될 경우, 호출 또는 노출 빈도가 높아 가중치가 부여된 검증 항목에 대해서는 그렇지 않은 검증항목보다는 우선순위가 높아 전체 범위 회귀 테스트 시 우선순위를 구분하여 테스트를 수행할 수 있다.



〈Figure 7〉 Flow Chart of Test Cases Weights and Prioritization

<Figure 7>은 메서드별 가중치를 부여하고, 사용자 활동 로그의 노출 빈도 및 호출 빈도로 우선순위를 부여하고, 코드 복잡도를 활용하여 검증 항목별 우선순위 정보가 저장되고 있는 과정을 도식화한 내용이다.

<Table 3>은 사용자 활동 로그 평가 항목별 가중치의 영향도를 최상, 상, 중, 하, 최하로 분류하고 가중치를 '최상'은 5, '상'은 4, '중'은 3, '하'는 2, '최하'는 1로 정의하였다. 단, 가중치는 호출 빈도와 노출 빈도 중 하나라도 조건에 부합할 경우, 가중치를 부여한다.

〈Table 3〉 Weights for User's Activity Log Evaluation Cases

Consistency	Conditions	Effect	Weight
Consistent with changing/ changed method	Call/exoposure frequency higher than 71%	Very high	5
	Call/exoposure frequency higher than 31%, lower than 70%	High	4
	Call/exoposure frequency higher than 0%, lower than 30%	Average	3
Inconsistent with changing/ changed method	Call/exoposure frequency higher than 51%	Low	2
	Call/exoposure frequency higher than 0%, lower than 50%	very low	1

사용자 중심의 활동 로그를 이용한 검증 항목의 우선순위를 적용 하는 방안을 5가지 항목으로 구분하고, 우선순위 척도의 세분화를 위해 코드 복잡도를 이용하여 2차적으로 우선순위를 보정한다. <Table 4>는 코드 복잡도별 가중치의 영향도를 최상, 상, 중, 하, 최하로 분류하고 가중치를 '최상'은 5, '상'은 4, '중'은 3, '하'는 2, '최하'는 1로 정의하였다(So and Chae, 2005).

〈Table 4〉 Weights for Codes complexities

Consistency	Conditions	Effect	Weight
Consistent with changing/ changed method	Method's complexity higher than 6	Very high	5
	Method's complexity higher than 3, lower than 5	High	4
	Method's complexity lower than 2	Average	3
Inconsistent with changing/ changed method	Method's complexity higher than 6	Low	2
	Method's complexity lower than 5	very low	1

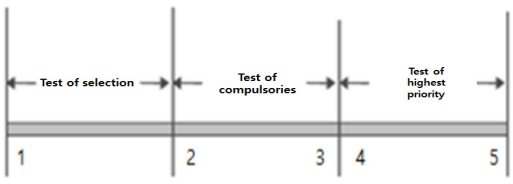
<Table 3> 사용자 활동 로그 가중치, <Table 4> 코드 복잡도 가중치를 합산하여 우선순위 지표로 사용한다.

회귀 테스트 시, 변경된 코드에 영향 받는 모든 기능에 대해 테스트를 수행해야 사용자가 안정된 품질의 제품을 사용할 수 있다. 그러나 시간적 제약 때문에 변경된 기능과 그로인해 영향 받는 주요 핵심 기능에 대해서만 검증을 하고 출시를 해야 하는 경우도 발생한다(Marnie, 2003). 이런 상황에서 사용자가 느끼는 결함을 최소화하기 위해 사용자 활동 로그를 활용하고 코드 복잡도를 통한 검증 범위의 우선순위를 조정할 수 있다. 우선순위는 아래 공식으로 사용하며, 소수 첫째 자리까지 표현하고 반올림하지 않고 그대로 사용한다.

우선순위 = (메서드 호출관계+사용자 활동 로그 +코드 복잡도)/3

각 가중치별 최고치는 메서드 호출관계 5, 사용자 활동 로그 5, 코드 복잡도 5를 합산한 15으로, 세 가지 지표의 가중치 합을 3으로 나누면 5로 우선순위가 부여되고 최우선 검증 항목으로 추출된다.

<Figure 8>에서는 메서드별 호출관계 가중치와 사용자 활동 로그 가중치의 합과 코드복잡도의 가중치를 가지고 3개의 등급과 각 등급 범위 값을 보여주고 있다. '최우선 검증' 항목의 경우 가중치 평가 결과가 4 이상에서 5의 값을 가지고 되고, '필수 검증' 항목은 2 이상에서 3 이하의 가중치 평가 값을 가지게 된다. 그리고 '선택 검증' 항목은 가중치 평가 값이 2 미만의 값을 갖게 된다(So and Chae, 2005).



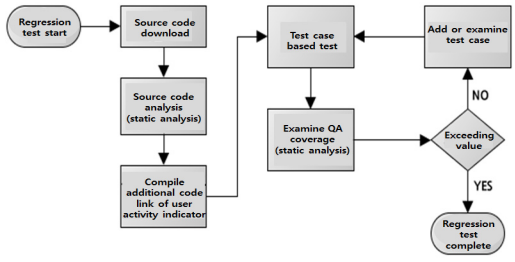
〈Figure 8〉 Example of Test Case Selection

<Figure 8>과 같이 전체 검증 항목에 대해 가중치를 부여하여 우선순위를 조정한 뒤 품질보증 담당자는 개발 담당자와 협의하여 최종적으로 선택 검증 항목에 대한 가감을 결정해야 한다. 시간적 제약이 발생한 경우에도 최우선 검증 항목과 필수 검증 항목은 반드시 테스트가 이루어진 상태로 소프트웨어가 출시되어야 한다. 선택 검증 항목에 대해서도 변경된 기능에 영향을 받는 기능으로, 테스트 진행 여부를 판단할 때 소프트웨어 품질을 높이기 위해 제외 범위를 최소화한다.

3.3 테스트 커버리지 향상

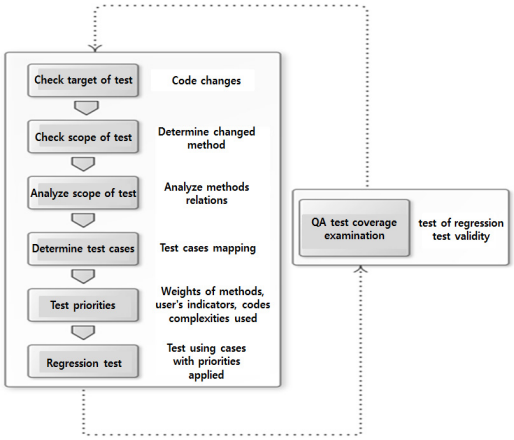
테스트 커버리지 측정은 테스트 결과를 분석하여 테스트가 충분한지를 평가하는 작업으로, 소스 코드의 변경 전과 후를 비교하여 테스트 커버리지 차이를 파악해야 한다. 소스 코드 변경 후 커버되지 않는 부분에 대해서는 검증 항목을 보완해야 한다(Willliam, 2006).

<Figure 9>의 테스트 커버리지 측정 시, 개발 담당자와 품질보증 담당자가 상호 협의한 기준을 초과할 때까지 검증 항목을 추가 또는 수정한다.



〈Figure 9〉 Test Coverage Examination Procedure

그러나 정적 분석을 활용하여 코드 커버리지 측정과 동일하게 품질보증 테스트 커버리지 측정이 가능하다. 각각의 메서드 정보와 기능 정의서가 맵핑되어 있고, 해당 기능에 대한 검증 항목 셋을 가지고 있다는 전제하에 코드 커버리지 측정과 동일하다.



〈Figure 10〉 Regression Test Process

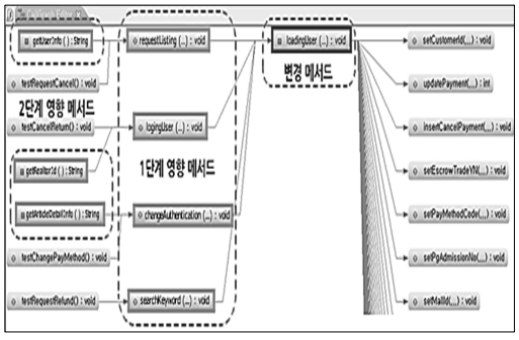
<Figure 10>은 본 연구에서 제안하는 효율적인 회귀 테스트 방법인 변경 중심 테스팅과 검증 항목의 우선순위 부여하는 과정을 보여주고 있다. 코드 변경이 발생하면 변경된 코드의 변경 메서드를 파악하고 영향 받는 메서드 관계를 분석한다. 정적 분석을 통해 메서드별 관계가 파악되면 검증 항목과 맵핑하여 검증 대상과 검증 항목을 분류한다. 분류 작업이 완료되면 회귀 테스트 시 검증 항목의 우선순위를 부여하기 위해 메서드별 호출관계와 수집된 사용자의 활동 로그를 분석한다. 분석된 사용자 활동 로그 지표를 활용해서 우선순위 가중치를 분류하고, 최종적으로 코드 복잡도를 활용하여 가중치 척도를 세분화하는 과정을 거친다.

회귀 테스트의 여러 우선순위 화기법 중 가장 비용 절감 효과가 있는 것은 테스트 항목 우선순위 화 기법이다(So and Chae, 2005). 그리고 그 기법을 본 논문에서는 메서드별 호출관계, 사용자 활동 로그, 코드 복잡도를 이용하여 우선순위를 부여하는 방법으로 제안했다. 검증 대상과 범위를 도출하고 해당 검증 항목의 우선순위에 따라 회귀 테스트를 수행하게 되면 적은 비용과 시간으로 효과적인 회귀 테스트를 할 수 있다.

추가적으로 회귀 테스트 수행 시 품질보증 테스트 커버리지 측정도 같이 되어, 테스트 진척도 및 수행 완료 범위에 대해 쉽게 파악할 수 있다는 장점이 있다. 검증 항목은 기능과 맵핑되어 있고 해당 기능은 메서드 정보를 가지고 있으므로, 메서드별 테스트 유무를 파악하기 용이하다.

4. 실험 및 결과

본 연구에서 정적 분석을 통한 회귀 테스트의 검증 범위를 도출하는 방안과 사용자 활동 로그를 활용한 검증 항목의 우선순위 부여를 통한 검증 항목을 가감하는 방안을 제안한다. 이를 실제 개발 중인 모바일 웹의 일부 기능에 적용함으로써, 품질보증 테스트 커버리지 향상 및 테스트 진척도 측정, 효과적인 결함 발견 활동에 대한 실험 및 결과를 기술한다(Naver, 2014).



〈Figure 11〉 Call Graph Creation Through Static Analysis

<Figure 11>은 변경사항 모니터링 모듈에 의해 개발자의 소스 커밋을 실시간 코드 변경 감지로 수정이 발생한 메서드에 영향 받는 메서드들을 정적 분석을 이용하여 call graph로 파악한 그림이다.

검증 항목 맵핑 모듈에 의해 정적 분석 후 생성된 call graph를 이용하여 변경 및 영향 받는 메서드에 해당하는 검증 항목을 저장된 회귀 테스트 전체 검증 항목에서 추출한다. 변경된 메서드별 기능에 해당하는 검증 항목과 맵핑하는 작업을 거친다. 이때 가중치는 메서드의 유형에 따라 변경된 메서드, 1단계 영향 메서드, 2단계 영향 메서드로 구분되며, 각각의 가중치는 자동으로 부여되게 된다. 3단

계 이상 영향 메서드는 전체 회귀 테스트 시 검증 대상에는 포함될 수 있으나, 본 연구에서는 결함이 발생할 확률이 높은 메서드인 변경 메서드, 1단계 영향 메서드, 2단계 영향 받는 메서드에만 가중치를 부여한다.

<Figure 12>는 사용자 활동 로그를 수집하여 각 영역별 노출 빈도와 호출 빈도를 보여주는 화면이다. 로그 분석 시스템을 통해 클릭 로그와 노출 로그를 분류하여 미리 지정한 영역에서 데이터를 수집하는 페이지 태깅 방식을 사용했다. 또한, 페이지 태깅(Page tagging) 방식을 이용하여 품질보증 담당자가 테스트를 수행할 때 로그를 남겨 실제 테스트 진척도 및 테스트 커버리지 측정에도 사용한다.

〈Figure 12〉 Log Analysis of Actual User's Activities

사용자 활동 로그 수집이 완료되면 맥케이브 순환 복잡도 메트릭(McCabe's complexity metric)을 이용하여 검증 항목의 우선순위 지표를 추가된다(Zhang et al., 2007).

<Figure 13>은 각 메서드별 코드 복잡도를 맥케이브 복잡도 메트릭을 이용하여 나타낸 로그 파일이다. 로그 파일의 최상단에는 변경 또는 영향 받는 메서드가 위치되도록 정렬하였다. 사용자 활동 로그 가중치 부여가 완료되면 최종적으로 코드 복잡도 가중치를 이용하여 우선순위를 적용한다.

	A	B	C	D	E	F	G
1	Target	FileType	File	Function	Complexity	TotalLoc	CodeLoc
2	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.loadingUser	8	34	27
3	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.requestListing	17	127	105
4	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.logingUser	3	40	23
5	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.changeAuthentication	16	107	65
6	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.searchKeyword	7	37	32
7	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getUserInfo	8	32	24
8	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getArticleDetailInfo	4	39	21
9	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getRealtorId	2	18	14
10	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getMember_type	1	3	3
11	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setMember_type	2	7	7
12	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getMbr_id	1	3	3
13	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setMbr_id	2	7	7
14	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getRlet_type_cd	1	3	3
15	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setRlet_type_cd	2	7	7
16	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getTrad_tp_cd	1	3	3
17	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setTrad_tp_cd	2	7	7
18	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getAtcl_rgst_prod_tp_cc	1	3	3
19	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setAtcl_rgst_prod_tp_cc	2	7	7
20	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getAtcl_stat_cd	1	3	3
21	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setAtcl_stat_cd	2	7	7
22	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getCity	1	3	3
23	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setCity	2	7	7
24	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getDvsn	1	3	3
25	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setDvsn	2	7	7
26	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getSec	1	3	3
27	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setSec	2	7	7
28	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getComplex	1	3	3
29	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.setComplex	2	7	7
30	20140415	Java	admin₩trunk₩src₩java₩	AbstractArticleListAction.getS_atcl_no	1	3	3

〈Figure 13〉 Code Complexity Analysis

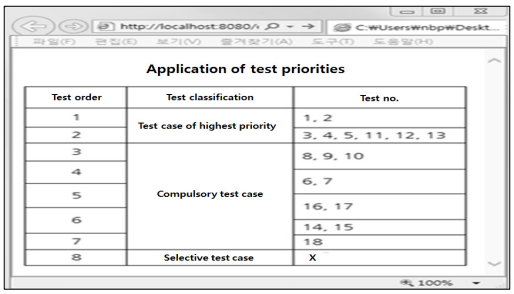
No.	Test cases	Weights for method	Weights for activity log	Weights for code complexity	Weights in average
1	관심 매물	5	4	5	4.6
2	관심 등록	5	4	5	4.6
3	지도 매물 노출	3	5	5	4.3
4	매물 유형별 노출	3	5	5	4.3
5	거래 유형별 노출	3	5	5	4.3
6	중개업소 아이디	3	3	4	3.3
7	개인 아이디	3	3	4	3.3
8	공인인증서 인증	3	3	5	3.6
9	휴대폰 인증	3	3	5	3.6
10	이메일 인증	3	3	5	3.6
11	통합 검색	3	5	5	4.3
12	매물 검색	3	5	5	4.3
13	서비스 내 검색	3	5	5	4.3
14	개인 회원	1	3	5	3
15	단체 회원	1	3	5	3
16	매물 종류	1	4	4	3
17	거래 종류	1	4	4	3
18	중개업소 위치	1	3	3	2.3

〈Figure 14〉 Application of Priorities

<Figure 14>는 검증 항목별 검증 우선순위를 부여하기 위해 세 가지 지표를 활용하여 가중치를 부여한 결과이다. 실험 데이터는 메서드별 세부적으로 기능 구현이 되어 있지 않고 공통 메서드가 다수 포함되어 있어 검증 항목별 유사한 가중치가 나왔지만, 일반적으로 공통 메서드를 제외하고는 세부적으로 가중치가 계산되어 나올 것으로 예상된다.

검증 항목별 실험 데이터의 최종 검증 항목 수행 순서는 최우선 검증 항목부터 필수 검증 항목으로 파악되어 선택 검증 항목 없이 모두 우선순위가 높은 것으로 조사되었다. 선택 검증 항목이 포함되어 있지 않으므로 개발 담당자와 협의 없이 모

든 검증 항목이 회귀 테스트 범위이며, 테스트 수행 전 검증 항목별로 메서드 가중치, 사용자 활동 로그 가중치, 코드 복잡도 가중치를 기준으로 분석 모듈에 의해 검증 항목의 검증 수행순서가 부여된다.



Application of test priorities

Test order	Test classification	Test no.
1	Test case of highest priority	1, 2
2		3, 4, 5, 11, 12, 13
3	Compulsory test case	8, 9, 10
4		6, 7
5		16, 17
6		14, 15
7		18
8	Selective test case	X

〈Figure 15〉 Application of Test Priorities

<Figure 15>는 메서드별 가중치 및 우선순위 부여 모듈에 의해 <Figure 14>의 가중치 합계에 따른 회귀 테스트 수행순서가 최종적으로 결정된 화면이다.

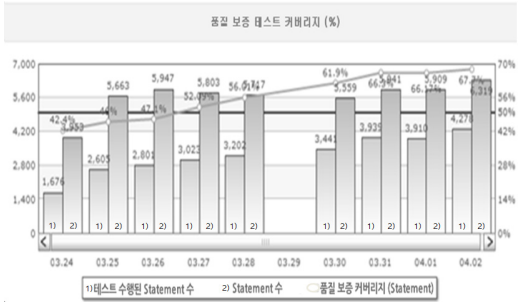
본 논문에서는 검증 범위와 검증 항목의 유효성을 확인하기 위해 회귀 테스트 수행 시 정적 분석이 가능한 코드를 심어 품질보증 커버리지 측정을 했다. 우선순위 부여가 완료되면 회귀 테스트를 수행하면서 진척도 측정과 각 메서드별 품질보증 커버리지 확인 후, 중요한 프로젝트의 경우 일별 진척도 확인 및 각 메서드별 테스트 진행정도를 파악해서 테스트 커버리지가 낮은 메서드는 재확인 과정을 거치거나 테스트 방법을 변경하는 작업을 할 수 있다.

<Figure 16>은 정적분석에 의한 유닛 테스트 커버리지와 회귀 테스트를 수행하면서 확인한 품질보증 커버리지를 보여주는 화면이다. 상단 점선 영역의 메서드들은 실험 예제의 변경된 메서드와 영향 받는 메서드에 해당한다. 테스트가 진행 중인 상태인데 대부분의 메서드에 대해서는 유닛 테스트 커버리지 보다 많은 부분이 검증되어 커버리지가 높아졌음을 확인할 있다(Ikeda et al., 2014).

<Figure 17>은 프로젝트 진행 중 회귀 테스트의 진척도 및 메서드별 테스트 커버리지를 주기적으로 확인하기 위해 품질보증 테스트 커버리지를 측정한 화면이다. 회귀 테스트를 수행하면서 특정 기능에 해당하는 메서드가 호출되면 커버리지가 증가하는 방식으로 추출된 검증 항목의 유효성을 확인할 수 있다. 변경된 메서드와 영향 받는 메서드에 해당 하는 검증 항목으로 테스트를 수행할수록 커버리지가 증가함을 보여주고 있다. 품질보증 커버리지 측정을 대규모 프로젝트에서 적용하면 회귀 테스트 종료일을 예측하거나 프로젝트 진척도의 일별 확인이 가능하여 프로젝트 관리 요소로 활용할 수 있다.

유닛 테스트 커버리지		품질 보증 커버리지		비고
AllTests	0%			Unit Test용 Class
loadingUser	75.50%	loadingUser	82.70%	▲
requestListing	77.30%	requestListing	77.30%	-
logingUser	84.60%	logingUser	100%	▲
changeAuthentication	90.80%	changeAuthentication	80%	▼
searchKeyword	0%	searchKeyword	0%	-
getUserInfo	93.50%	getUserInfo	93.50%	-
getArticleDetailInfo	67.70%	getArticleDetailInfo	91.30%	▲
getRealtorId	0%	getRealtorId	40.80%	▲
ChargeCrosCheckTestMain.ChargeData	0%			Unit Test용 Class
ChargeDailyCorrection	78.40%	ChargeDailyCorrection	77.80%	▼
ChargeDao	52.30%	ChargeDao	48.20%	▼
ChargeData	86%	ChargeData	91.30%	▲
ChargeDataFetcher	65.50%	ChargeDataFetcher	78.90%	▲
ChargeGroupData	36.80%	ChargeGroupData	40.80%	▲
ChargeTestUtils	80.30%			Unit Test용 Class
ChargeTestUtils.GroupInfo	76.50%			Unit Test용 Class
ChargeUtils	100%	ChargeUtils	81.20%	▼
CheckDailyBudgetJob	0%	CheckDailyBudgetJob	0%	-
CpcChargeData	73.70%	CpcChargeData	73.70%	-
CpcJobInvoker	43.20%	CpcJobInvoker	37.80%	▼
HealthCheck	0%	HealthCheck	61.50%	▲
JobService	0%	JobService	53.60%	▲
JobServiceParams	41.20%	JobServiceParams	68.60%	▲
JobServiceTask	0%	JobServiceTask	96.70%	▲
Main	0%	Main	66.70%	▲
SMSBroker	37%	SMSBroker	0%	▼
ServerInfo	0%	ServerInfo	0%	-
ServiceExecutorMain	0%			Unit Test용 Class
Sleep	0%	Sleep	0%	-
SqlMapClientFactory	88.90%	SqlMapClientFactory	88.90%	▼
TestDBUtil	0%			Unit Test용 Class
VerifyChargeAmtJob	0%	VerifyChargeAmtJob	0%	-
전체 Coverage	52.90%	전체 Coverage	57.90%	

〈Figure 16〉 QA Coverage Examination



〈Figure 17〉 Regression Test Coverage-Daily Progress

회귀 테스트 시 주로 사용하는 변경 기능 및 영향 받는 기능에 대한 품질보증 담당자의 정성적인 판단으로 테스트하는 방법과 본 논문에서 제안하는 회귀 테스트 방법을 우선순위 부여, 결함 발견, 수행시간 예측, 검증 항목 유지 보수 등 주요 항목으로 비교하면 <Table 5>의 적용 전후 내용을 확인할 수 있다.

본 실험에서는 회귀 테스트의 유효성을 검증하기 위해 주기적인 품질보증 테스트 커버리지 측정을 하여 변경된 소프트웨어가 릴리즈(Release) 가능한 수준으로 검증이 되었는지 판단하는 평가 기준이 되었고, 결함이 발생할 확률이 가장 높은 변경된 코드의 메서드와 영향 받는 메서드에 대해 테스트를 집중적으로 진행하여 품질 안정화의 효율성을 높였다. 기존의 회귀 테스트 방법과의 비교 결과 <Table 5>의 우선순위 부여 방법, 결함 누락 발견, 수행 시간 예측, 검증 항목 유지 보수의 항목에서 모두 효과가 있음을 확인하였다.

5. 결론 및 향후 연구 과제

소프트웨어는 사용자의 요구사항에 따라 복잡해지고 지속적으로 변하게 된다. 이러한 소프트웨어의 변경에 따른 품질보증을 위해 회귀 테스트 단계를 통해 결함을 발견하고 수정하는 과정을 거친다. 테스트 수행 중 신규로 발견된 결함을 수정하는 과정에서 새롭게 유입되는 결함이 있는지도 확인하는 반복 테스트가 필요하다. 그러나 소프트웨어의 기능이 추가될수록 실행해야 하는 검증 항목도 늘어나기 때문에 선택적 회귀 테스트 기법을 이용하게 된다.

본 논문에서 제안한 검증 범위 선정과 검증 항목 추출 및 우선순위 부여방법은, 변경된 프로그램의 효과적인 테스트와 검증 유효성을 확인하는데 활용한다. 제안한 회귀 테스트 방법은 4단계로 이루어진다. 첫 번째 단계에서는 변경된 프로그램을 정적 분석을 통해 메서드 흐름을 식별하여 검증 범위를 선정한다. 두 번째 단계에서는 변경된 부분 및 영향 받는 범위에 대한 회귀 테스트 시 사용할 검증 항목을 도출한다. 세 번째 단계에서는 도출된 검증 항목 중 검증 순서를 부여하기 위해 메서드별 중요도, 사용자 활동 로그, 코드 복잡도를 활용하여 최적화하는 과정을 수행한다. 네 번째로는 품질보증 커버리지 측정을 통한 변경 중심 검증 방법에 대한 유효성을 확인한다.

일반적인 회귀 테스트를 수행할 때와 본 논문에서 제안한 기법을 적용할 때의 이점은 소프트웨어의 변경으로 영향 받는 부분만을 식별하기 때문에 중요도가 낮은 검증 범위를 최소화할 수 있는 것이다. 검증 항목의 우선순위는 사용자 활동 로그와 코드 복잡도를 분석하여 검증 우선순위를 부여한다. 이렇게 우선순위가 부여된 검증 항목으로 검증을 순차적으로 수행함으로 시간과 비용을 절약할 수 있다.

본 논문은 품질보증 커버리지 측정으로 테스트

〈Table 5〉 Weights for Codes Complexities

Cases	Before	After
Application of Weights	Less objective due to the QA administrator's qualitative examination	More objective due to quantitative weights data from priority indicators
Possibility of dectection of effective faults or omission of faults	Faults can be omitted depending on the depth of QA administrator's domain knowledge	Minimized possibility of omission due to consistent test cases
Performance elapsed time prediction	Can't be predicted	Predictable due to previous test history
Test cases maintenance	Modification/addition on the sector of effect is difficult	Possible to determine the exact sector of effect-easier maintenance

진척도 확인 및 기능별 검증 유무를 확인할 수 있는 회귀 테스트의 효율성 향상에 관한 연구다. 논문에서 제안한 검증 방법은 정적 분석과 동적 분석의 장점을 모두 조합한 방법으로 효과적으로 소프트웨어의 결함을 발견할 수 있다. 그러나 코드 변경이 발생할 때마다 변경된 코드에 해당하는 검증 항목을 확인해서 검증 항목을 추가 또는 수정하는 과정이 필요하다. 이는 사람에 의한 수동적인 방법으로 누락 또는 실수가 발생할 확률이 있다. 코드 커버리지 측정이 어려운 기능 변경이 없는 쿼리 수정이나 데이터베이스의 구조 변경이 발생할 경우에는, 코드 변경을 인지할 수 없어 검증 범위 및 검증 항목을 도출하기가 어렵기 때문에 코드 변경과 더불어 데이터베이스의 변경 모니터링에 관한 연구가 추가적으로 필요하다.

References

Kim, S.D., *Software Engineering for Practitioners*, AdTech, 1999.

(김수동, *실무자를 위한 소프트웨어공학*, 에드텍 출판사, 1999.)

Kim, Y.W., "A Usability Analysis Method of Mobile Application using User Behavior Logs", M.A. Thesis, University of Sogang, Korea, 2004.

(김영욱, "의도 기반의 사용자 행위 로그를 이용한 모바일 어플리케이션의 사용성 분석 기법", 서강대학교 석사논문, 한국, 2004.)

Seo, J.H., "The Method to Deduct Cases of Priority Regression Test by Utilizing the Method Flow", M.A. Thesis, University of Sogang, 2013.

(서주희, "메서드 흐름을 활용한 우선순위 회귀 테스트 케이스 도출 방법", 서강대학교 정보통신대학원 석사논문, 2013.)

So, S.S. and Y.G. Chae, "Empirical Study on Test Case Prioritization Techniques of Regression Testing", *Korea Information Processing Society*, Vol.12, No.2, 2005, 283-288.

(소선섭, 채의근, "회귀 테스팅의 테스트 케이스 우선 순위화 기법의 실험적 연구", *정보처리학회논문지*, 제12권, 제2호, 2005, 283-288.)

Ikeda, D., H. Kazuaki, and I. Humiaki, *Developing Practical Skills to Lead Successful Team*, J.Pub, 2014.

(이케다 다카후미, 후지쿠라 카즈아키, 이노우에 후미아키, *성공으로 이끄는 팀 개발 실천 기술*, 제이펍, 2014.)

Lee, H.J. and W.I. Kwon, *Practical Software Testings for the Developers Third Edition*, STA, 2011.

(이현주, 권원일, *개발자도 알아야 할 소프트웨어 테스팅 실무 3판*, STA, 2011.)

Leung, H.K.N. and L. White, "Insights Into Regression Testing", S*oftware Testing, Software Maintenance*, 1989, 60-69.

Rothermel, G., R.H. Untch, C. Chengyun, and M. J. Harrold, "Prioritizing Test Cases for Regression Testing", *Software Engineering*, IEEE Transactions on, Vol.27, No.10, 2001, 929-948.

Mei, L., Z. Zhang, W.K. Chan, and T.H. Tse, "Test Case Prioritization for Regression Testing of Service-Oriented Business Applications", In *Proceedings of the 18th International Conference on World Wide Web,* 2009, 901-910.

Thomas, M.C., "Software Quality Metrics to Identify Risk", *Department of Homel and Security Software Assurance Working Group*, 2008

Tim, R. and G. Adam, *Beautiful Testing : Leading Professionals Reveal How They Improve Software*, O'Reilly Media, Inc, 2009.

Tip, F. and J. Palsberg, "Scalable propagation-

based call graph construction algorithms", *OOPSLA Proceedings of the 15th ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages, and Applications*, Vol.35, No.10, 2000, 281-293.

Zhang, H., X. Zhang, and M. Gu, "Predicting Defective Software Components from Code Complexity Measures", *Dependable Computing, PRDC 13th Pacific Rim International Symposium on*, 2007, 93-96.

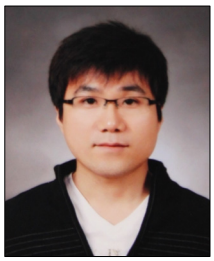
Marnie, L.H., *Software Testing Fundamentals : Methods and Metrics*, John Wiley and Sons, 2003.

Willliam, E.P., *Effective Methods for Software Testing 2nd Edition*, Wiley India Pvt. Limited, 2006.

Wikipedia, "Regression Testing Definition", 2014, Available at http://en.wikipedia.org/wiki/Regression_testing(Accessed Jan. 1).

Naver, "Real Property Mobile Web", 2014, Available at http://m.land.naver.com/(Accessed Jan. 1).

About the Authors



Woojin Jung (dave.jung@sk.com)

Woojin Jung received his master's degree in Software Engineering from Soongsil University in Seoul, Korea. Currently, he is working as a principal member of the quality assurance engineer in SK Telecom Product Development Team. His main research interests include testing modeling, testing coverage, software engineering, and the performance analysis of cloud computing.

Sang-Rin Rah (nasangrin@naver.com)

Sang-rin Rah has received the degree in Applied Mathematics from Soongsil University in 2014. He is currently taking master's course in Software at Graduate School of Software in Soongsil University. His current research topics include big data, regression testing, open source architecture, data modeling, data analysis, software engineering.

Yong-Lak Choi (ylchoi58@ssu.ac.kr)

Professor Yong-Lak Choi received Ph.D. degree in Computer Science from Graduate School of Soongsil University in 2002. He is currently a professor of Graduate School of Software of Soongsil University. His research and main interest areas are databases, data modeling, big data, software engineering, information strategy planning, and open source architecture.