ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 43, No. 8, pp. 883-892, 2016. 8
http://dx.doi.org/10.5626/JOK.2016.43.8.883

결함위치식별 기법의 성능 향상을 위한 테스트케이스 그룹화 및 필터링 기법

(Test Case Grouping and Filtering for Better Performance of Spectrum-based Fault Localization)

김 정 호 †　　이 은 석 ††
(Jeongho Kim)　　(Eunseok Lee)

요 약 스펙트럼 기반 결함위치식별 기법은 성공 테스트케이스 대비 실패 테스트케이스에 영향을 많이 받은 스테이트먼트에 의심도를 통계적으로 부여하는 기법이다. 이 기법에서 실패 테스트케이스를 지나간 스테이트먼트에는 의심도를 부여하고 성공 테스트케이스를 지나간 스테이트먼트는 부여된 의심도 일부를 줄이는 역할을 한다. 그러므로 실패 테스트케이스의 역할이 매우 중요하며 부재 시 결함의 위치를 찾을 수 없기도 하다. 하지만 이 기법에서 실패, 성공 테스트케이스를 동시에 입력하여 의심도를 계산하기 때문에 실패 테스트케이스의 고유 특성을 반영할 수 없다는 한계점을 가지고 있다. 본 논문에서는 이와 같은 한계점을 보완하여 보다 정확한 결함위치식별을 도와줄 수 있는 테스트케이스 그룹화기법을 제안한다. 또한, 테스트 효율성을 고려한 필터링 기법을 제안하고 이들을 65개의 알고리즘에 적용해 실효성을 검증한다. EXAM score기준으로 전체의 90% 기법에서 정확도 13%, 효율성이 72% 향상되었다.

키워드: 스펙트럼기반 결함위치식별, 테스트케이스 그룹화, 필터링, 프로그램 실행 정보, 의심도, EXAM score

Abstract Spectrum-based fault localization (SFL) method assigns a suspicious ratio. The statement is strongly affected by a failed test case compared to a passed test case. A failed test case assigns a suspicious ratio while a passed test case reduces some parts of assigned suspicious ratio. In the absence of a failed test case, it is impossible to localize the fault. Thus, a failed test case is very important for fault localization. However, spectrum-based fault localization has difficulty in reflecting the unique characteristics of a failed test because a failed test case and a passed test case are input at the same time to calculate a suspicious ratio. This paper supplements for this limitation and suggests a test case grouping method for more accurate fault localization. In addition, this paper suggested a filtering method considering test efficiency and verified the effectiveness by applying 65 algorithms. In 90 % of whole methods, the accuracy was improved by 13% and the effectiveness was improved by 72% based on EXAM score.

Keywords: spectrum-based fault localization, test case grouping, filtering, program trace, suspicious ratio, EXAM score

· 이 논문은 2015년도 정부(교육과학기술부)의 재원으로 한국연구재단-차세대 정보컴퓨팅개발사업의 지원을 받아 수행된 연구임(No. 2015045358)
· 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음(R2215-16-1005)
· 이 논문은 2015 한국 소프트웨어공학 학술대회에서 '테스트케이스 재구성을 통한 결함위치식별 성능개선'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 성균관대학교 전자전기컴퓨터공학과
jeonghodot@skku.edu
†† 종신회원 : 성균관대학교 컴퓨터공학과 교수
(Sungkyunkwan Univ.)
leees@skku.edu
(Corresponding author임)

논문접수 : 2016년 3월 11일
(Received 11 March 2016)
논문수정 : 2016년 5월 12일
(Revised 12 June 2016)
심사완료 : 2016년 5월 17일
(Accepted 17 May 2016)

정보과학회논문지 제43권 제8호(2016. 8)

1. 서 론

소프트웨어에 대한 의존도가 높아져 감에 따라 양질의 고신뢰 소프트웨어를 생산하기 위한 각별한 노력이 요구된다. 고품질의 소프트웨어는 개발 기술(요구공학, 설계, 구현 등)에 의한 영향력 이상으로, 개발된 결과물에 대한 테스트 및 디버깅 기술이나 성능이 특히 중요하다. 개발된 결과물로서의 소스코드에 대한 결함 존재 유무를 확인하는 테스트의 경우, 이미 효율적인 많은 방법론과 지원도구가 개발되어 있다. 그러나 테스트 이후 코드의 결함 위치를 구체적으로 식별해줄 수 있는 기술은 기술적 난이도 등으로 인해 상대적으로 개선의 여지가 많이 남아있다. 이로 인해 여전히 산업계 및 학계에서의 높은 소프트웨어 시험부하는 소프트웨어의 품질은 물론 비용 상승에 큰 원인이 되고 있다[1,2]. 특히, 결함위치식별은 전체 소프트웨어 개발 비용의 50-80%를 차지하고 있으며[1], 디버깅을 위해 반드시 선행되어야하므로 매우 중요한 작업이다. 이러한 이유로 인해 최근까지도 많은 연구가 진행되고 있으며, 이들의 대부분은 CFG(Control-Flow Graph), DFG(Data-Flow Graph) 등과 같은 프로그램의 정보를 활용해 결함 위치를 식별한다. 특히, 결함위치식별에 가장 유용한 힌트를 제공하는 정보는 실패 테스트케이스의 실행정보이다. (실패, 성공 테스트케이스는 각각 테스트의 실제 결과가 예상된 결과와 다르거나 같은 것을 의미함)

그림 1에 소개된 결함위치식별 기법들은 일반적으로 결함의 위치를 찾기 위해 사용되는 기법들이다. 이들은 모두 실패 테스트케이스 정보를 기준으로 결함의 위치를 추적한다. 스펙트럼 기반(Spectrum-based) 기법은 실패, 성공 테스트케이스의 지나간 정보를 활용해 성공 테스트케이스 대비 실패 테스트케이스에 영향을 많이 받은 스테이트먼트에 의심도를 통계적으로 부여하는 기법이다[4-13]. 이 기법에서 실패 테스트케이스를 지나간 스테이트먼트에는 의심도를 부여하고 성공 테스트케이스를 지나간 스테이트먼트는 부여된 의심도 일부를 삭감하는 역할을 한다. 그러므로 실패 테스트케이스의 역할이 매우 중요하며 실제로 실패 테스트케이스의 부재시 결함의 위치를 찾을 수 없다[3].

델타 디버깅(Delta debugging) 기법은 결함의 원인이 되는 부분의 범위를 점점 좁혀가면서 최소한의 집합을 찾기 위한 기법이다[14,15]. 이 기법에서 기준점이 되는 역할을 하는 것이 실패 테스트케이스이고 반드시 최소 하나는 존재해야 적용이 가능하다.

변형 프로그램 기반(Mutation-based) 기법은 실패 테스트케이스가 지나간 스테이트먼트 일부를 수정해 테스트 결과가 변하는지를 확인하고, 정도에 따라 의심도

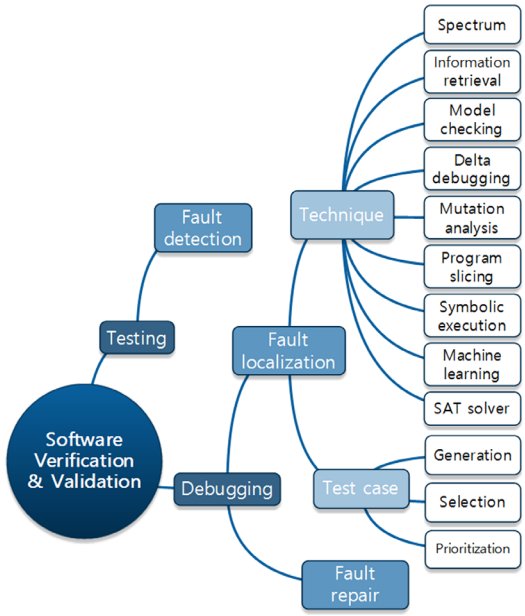

그림 1 소프트웨어 Verification & Validation
Fig. 1 Software Verification & Validation

를 부여하는 기법이다. 이 기법에서 실패 테스트케이스의 트레이스(trace) 정보는 수정을 시도할 소스코드의 모집단으로 선택되므로 필수 조건이다[16-18].

충족 가능성 기반(SATisfiability-based) 기법은 초기에 BMC(Bounded Model Checker)로 반례(Counter-example)를 찾고 이를 지나간 스테이트먼트의 소스코드를 제약사항으로 추출한다. SAT solver를 활용해 이들 간의 상호 모순 관계가 있는 스테이트먼트를 찾는 기법이며 반례를 실패 테스트케이스와 동일하다고 볼 수 있다[19-22]. 상기 기법들을 포함한 대부분의 기법에서 실패 테스트케이스는 결함의 위치를 추적하기 위해 매우 중요한 역할을 한다. 하지만 이 정보의 중요성을 다소 간과하는 일부 기법들이 존재한다. 본 논문에서는 SFL(Spectrum-based Fault Localization) 기법의 이러한 한계점을 보완하여 결함위치식별의 정확도와 효율성을 향상시킨다.

스펙트럼 기반 기법에서도 테스트케이스의 정보를 활용해 의심도를 도출해내는 과정에서 실패 테스트케이스에 가장 높은 가중치를 부여한다. Wong 기법[7]은 의심도를 실패 테스트케이스의 수로 정의해 각 스테이트먼트가 몇 개의 실패 테스트케이스가 실행 되었는지에 따라 의심도를 부여한다. D star 기법[8]는 Kulczynski 기법에 실패 테스트케이스의 수를 거듭제곱함으로써 실패 테스트케이스에 가중치를 부여하는 기법이다. 또한, Heurstic 기법[9]는 실패 테스트케이스의 반영비율을 사

전에 경험적으로 설정한 기준에 따라 의심도를 부여하는 기법이다. Naish 기법[10] 는 실패 테스트케이스의 수만큼 의심도를 부여하고 성공 테스트케이스가 지나간 비율로 일부를 정제하는 기법이다.

상기 기법과 같이 실패 테스트케이스 활용의 유의미함은 기 확인 하였으며, 실제로 다수 결함위치식별 기법들이 이 정보를 이미 활용하고있다. 하지만 실패, 성공 테스트케이스를 동시에 입력해서 의심도를 계산한다는 것 자체로는 실패 테스트케이스의 고유 특성을 반영하기 어렵다는 한계점을 가지고있다.

이러한 이유로 인해, 새로운 스펙트럼 기반 기법 제안보다는 테스트케이스 그룹화 및 필터링 기법 제안을 통해 테스트 정확성 및 효율성을 향상시키고자 한다. 본 논문의 기여는 다음과 같다.

(1)실패 테스트케이스의 수만큼 그룹화하여 각각의 의심도를 계산하고 누적하는 테스트케이스 그룹화 기법을 제안해 정확성을 향상시킨다.

(2)실패 테스트케이스를 적절하게 정제해줄 수 있는 일부 성공 테스트케이스만 선택하는 테스트케이스 필터링 기법을 제안해 정확성과 효율성을 향상시킨다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 소개하고 3장에서 테스트케이스 그룹화 및 필터링 기법을 제안한다. 4장에서 실험 계획과 결과를 설명하고 5장에서 본 논문에서 제안한 기법의 강, 약점에 대해 논의한다. 마지막으로 6장에서 결론을 내리고 7장에서 향후 연구에 대해 소개한다.

2. 관련 연구

2.1 스펙트럼 기반 결함위치식별 기법

스펙트럼 기반 기법은 프로그램의 실행 정보와 테스트케이스의 결과를 활용해 의심스러운 스테이트먼트를 식별하는 기법이다. 통계적인 접근을 통해 의심도를 계산하고 이를 기반으로 디버깅을 위해 검사해야 할 스테이트먼트의 순서를 결정해 주는 기법이다. 실패한 테스트케이스는 스테이트먼트에 의심도를 부여하고 성공한 테스트케이스는 부여된 의심도에서 결함과 관련이 없는 스테이트먼트의 의심도 일부를 줄이는 역할을 한다. 이 기법의 특성상 실제 결함이 존재하는 스테이트먼트를 지나가지 않는 테스트케이스는 모두 성공하며, 실패한 테스트케이스가 적어도 하나 이상 존재하지 않으면 결함의 위치를 찾을 수 없다.

표 1은 스테이트먼트 실행 여부와 테스트 결과를 기준으로 4가지 종류의 변수로 나타낸다. N_{cf} 는 스테이트먼트가 지나가고 테스트케이스의 결과가 실패인 경우, N_{uf} 는 스테이트먼트가 지나가지 않고 테스트케이스의 결과가 실패인 경우를 의미한다. N_{cf} 는 스테이트먼트가 지나가고 테스트케이스의 결과가 성공인 경우, N_{uf} 는 스테이트먼트가 지나가지 않고 테스트케이스의 결과가 성공인 경우를 의미한다.

표 1 의심도 계산을 위한 변수
Table 1 Parameters for calculating suspicious ratio

		Statement cover	
		Yes	No
Test result	Fail	N_{cf}	N_{uf}
	Pass	N_{cp}	N_{up}

표 2 스펙트럼 기반 대표 유사도측정기법
Table 2 Representative algorithms for spectrum-based fault localization

Similarity	Formulas
Tarantula	$\frac{\frac{N_{cf}}{N_{cf}+N_{uf}}}{\frac{N_{cf}}{N_{cf}+N_{uf}}+\frac{N_{cp}}{N_{cp}+N_{up}}}$
Ochiai	$\frac{N_{cf}}{\sqrt{(N_{cf}+N_{uf}) * (N_{cf}+N_{cp})}}$
Jaccard	$\frac{N_{cf}}{N_{cf}+N_{uf}+N_{cp}}$
AMPLE	$\left\lvert\frac{N_{cf}}{N_{cf}+N_{uf}}-\frac{N_{cp}}{N_{cp}+N_{up}}\right\rvert$

스펙트럼 기반 기법은 표 1에서 설명한 4가지 변수를 구하고 유사도측정기법 활용해 의심도를 계산한다. 대표 기법으로는 Tarantula, Ochiai, Jaccard, AMPLE [4-6, 8-13] 등이 있고 수식은 표 2에서 설명한다.

표 2에서 설명한 수식 외에도 일반적으로 유사도 또는 거리 측정을 목적으로 고안된 모든 기법을 적용할 수 있다. 또한, 이 외에도 결함위치식별을 위해 고안된 $\text{Lex}_{\text{Ochiali}}$[1], Naish1, 2[10], SEM1, 2, 3[4], Wong1, 2, 3[10], GP1-30[23], Heuristic1, 2, 3[9], D star[8]와 같은 경험적 기법들도 다수 존재한다.

표 1에서의 N_{cf}, N_{uf}, N_{cp}, N_{up} 와 표 2에서의 유사도측정기법을 활용해서 표 3에서 설명하고 있는 스펙트럼 기반 결함위치식별 기법의 의심도를 계산할 수 있다. 우선 테스트케이스의 실행 정보와 테스트 결과를 기반으로 개별 스테이트먼트의 N_{cf}, N_{uf}, N_{cp}, N_{up} 를 계산한다. 계산된 변수의 값을 유사도측정기법에 대입하면 개별 스테이트먼트 별로 의심도를 계산할 수 있다. 6번 스테이트먼트를 예로 들면, 전체 테스트케이스 6개 중 성공한 테스트케이스는 5개, 실패한 테스트케이스는 1개이다. 성공한 테스트케이스 중 6번 스테이트먼트를 지나가는 테스트케이스는 2개이므로, N_{cp} = 2가 되고, 그 전체에서 N_{cp} 를 제외한 나머지인 3이 N_{up} 가 된다. 또한 실

표 3 스펙트럼 기반 결함위치식별 기법의 의심도 계산 예시
Table 3 Example for calculating suspicious ratio using spectrum-based fault localization methods

	Test case						Algorithms			
mid() { int x,y,z,m;	3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3	Tarantula	Jaccard	Ochiai	AMPLE
1: read("Enter 3 numbers:",x,y,z);	●	●	●	●	●	●	0.50	0.17	0.41	0.00
2: m = z;	●	●	●	●	●	●	0.50	0.17	0.41	0.00
3: if (y<z)	●	●	●	●	●	●	0.50	0.17	0.41	0.00
4: if (x<y)	●	●			●	●	0.63	0.25	0.50	0.40
5: m = y;		●					0.00	0.00	0.00	0.20
6: else if (x<z)	●				●	●	0.71	0.34	0.58	0.60
7: m = y; // *** bug ***	●					●	0.83	0.50	0.71	0.80
8: else			●	●			0.00	0.00	0.00	0.40
9: if (x>y)			●	●			0.00	0.00	0.00	0.40
10: m = y;			●				0.00	0.00	0.00	0.20
11: else if (x>z)				●			0.00	0.00	0.00	0.20
12: m = x;							0.00	0.00	0.00	0.00
13: print("Middle number is:",m);	●	●	●	●	●	●	0.50	0.17	0.48	0.00
} Pass/Fail Status	P	P	P	P	P	F	Test case result			

Suspicious ratio

패한 테스트케이스 중 6번 스테이트먼트를 지나가는 테스트케이스는 1개이므로 $N_{cf}=1$이 되고, 그 전체에서 N_{cf}를 제외한 나머지인 0가 N_{uf}가 된다. 그러므로 N_{cf}, N_{uf}, N_{cp}, N_{up}는 각각 1, 0, 2, 3가 되고, 6번 스테이트먼트의 의심도를 계산하면 Tarantula는 약 0.71, Jaccard는 약 0.34, Ochiai는 약 0.58, AMPLE은 약 0.60이 된다.

3. 제안 기법

3.1 테스트케이스 그룹화 기법

결함위치식별의 정확성 향상을 위해 실패 테스트케이스의 수만큼 그룹을 만들고, 각 그룹의 의심도를 개별적으로 계산하여 누적하는 테스트케이스 그룹화 기법을 제안한다. 이를 통해 실패 테스트케이스의 가중치를 보다 높이고 고유 특성을 반영하여 정확성을 향상시킨다.

스펙트럼 기반 기법은 소스코드와 성공, 실패 테스트케이스가 주어지면, 개별 스테이트먼트를 지나가는 성공, 실패 테스트케이스의 수를 계산하고 이들을 유사도 또는 거리 계산 기법에 대입해 의심도를 구하는 기법이다.

알고리즘 1은 테스트케이스 그룹화 기법을 설명하고 있으며 기존 스펙트럼 기반 기법에 2, 9번 라인만 추가된 것이다. 2번 라인은 실패 테스트케이스의 수만큼의 그룹을 생성하여 계산을 그 수만큼 반복하라는 명령문이고, 9번 라인은 2번 라인의 반복문에서 계산되는 의심도를 모두 누적하여 통합하라는 명령어이다.

3.2 테스트케이스 필터링 기법

실패 테스트케이스는 의심도를 부여하는 역할을 하고 성공 테스트케이스는 부여된 의심도에서 결함과 다수 관련이 적은 부분의 의심도를 일부 삭감하는 역할을 한다. 대부분의 테스트케이스 조합에서는 일부 반복되는 테스트케이스를 포함하는 경우가 대부분인데 본 논문에서 제안한 테스트케이스 그룹화 기법 적용에는 적합하지 않다. 그러므로 본 장에서는 테스트케이스 필터링 기법을 제안해 중복된 테스트케이스를 관리한다.

알고리즘 2는 테스트케이스 필터링 기법을 설명하며, 성공, 실패 테스트케이스에 각각 선택적으로 적용할 수 있다. 중복된 테스트케이스가 존재하는지를 확인하고 존재한다면 해당 테스트케이스를 제거하고 처음 선택된 것이면 각 집합에 저장하는 기법이다. 이를 통해 테스트케이스는 유일무이한 집합으로 구성된다[4,24].

알고리즘 1 테스트케이스 그룹화 기법
Algorithm 1 Test case grouping method

NStmt = the Number of Statements
NFTC = the Number of Failed Test cases
NPTC = the Number of Passed Test cases

$S=\{s_1, s_2, s_3 \dots s_n\}$ = Statements
$F=\{f_1, f_2, f_3 \dots f_n\}$ = Failed test cases
$P=\{p_1, p_2, p_3 \dots p_n\}$ = Passed test cases

1: For (i=1, until NStmt){
2: For (j=1, until NFTC){
3: $N_{cf} = Count.coveredTC(Count.failedTC(s_i, f_j, P))$
4: $N_{uf} = Count.uncoveredTC(Count.failedTC(s_i, f_j, P))$
5: $N_{cp} = Count.coveredTC(Count.passedTC(s_i, f_j, P))$
6:
$N_{up} = Count.uncoveredTC(Count.passedTC(s_i, f_j, P))$
7: $s_{i_{suspicious\ ratio}}$ =Similarity$(N_{cf}, N_{uf}, N_{cp}, N_{up})$ *or*
Inverse(Distance$(N_{cf}, N_{uf}, N_{cp}, N_{up})$)
8: }
9: $s_{i_{suspicious\ ratio}}$ ***=Accumulate(***$s_{i_{suspicious\ ratio}}$***)***
10: }
11: $s_{i_{ranking}} = Rank(S, s_i, s_{i_{suspicious\ ratio}})$

알고리즘 2 테스트케이스 필터링 기법
Algorithm 2 Test case filtering method

NFTC = the Number of Failed Test cases
NPTC = the Number of Passed Test cases

$F=\{f_1, f_2, f_3 \dots f_n\}$ = Failed test cases
$P=\{p_1, p_2, p_3 \dots p_n\}$ = Passed test cases

Passed test case
1: For $(i = 1, until\ NPTC)$*{*
2: If $(p_i$ *is existed in* $P)$
3: $delete(p_i)$
4: Else
5: $input(p_i)$
6: }
7: Return P

Failed test case
8: For $(j = 1, until\ NFTC)$*{*
9: If $(f_j$ *is existed in* $F)$
10: $delete(f_j)$
11: Else
12: $input(f_j)$
13: }
14: Return F

4. 실 험

테스트케이스 그룹화 및 필터링 기법의 성능을 측정하기 위해 다수의 유사도 및 거리 측정 기법에 테스트케이스 그룹화 기법을 적용한다. 적용 기법은 Pearson, Euclid, Manhattan 등을 포함한 65개의 기법으로 설정한다.

4.1 평가 지표

스펙트럼 기반 기법들의 성능을 평가하기 위한 지표로써 소프트웨어 결함위치식별 분야에서 주로 사용하는 EXAM score를 활용한다[4,6,9,11-13,17].

$$EXAMscore = \frac{Ranking\ of\ actual\ faulty\ statement}{Total\ line\ of\ code} \times 100$$

EXAM score는 전체 소스코드 라인 수에 대한 실제 결함 위치의 순위를 백분율로 표현한 지표이다. 이는 실제 결함의 위치를 찾을 때까지 불필요하게 검사해야하는 코드의 양을 비율화한 개념이다. 예를 들어 200라인의 소스코드에서 실제 결함의 위치를 랭킹 3에서 찾았다면 EXAM score는 ((3/200)×100) = 1.5가 된다.

4.2 대상 프로그램

실험 대상 프로그램은 Siemens test suite[25-27]으로 선택한다. 이는 Software-artifact Infrastructure Repository[28]에서 각종 테스트를 위해 제공해주는 프로그램으로 소프트웨어 결함위치식별 분야에서 가장 많이 활용된다.

표 4 Siemens test suite
Table 4 Siemens test suite

Program	Faulty versions	Line of Code	Test cases	Description
Printtokens	7	472	4130	lexical analyzer
Printtokens2	10	399	4115	lexical analyzer
Replace	32	512	5542	pattern replacement
Schedule	9	292	2650	priority scheduler
Schedule2	10	301	2710	priority scheduler
Tcas	41	141	1608	altitude separation
Totinfo	23	440	1052	information measure

표 4의 Siemens test suite은 총 7개의 프로그램으로 구성되고 132개의 결함 버전과 17,677개의 테스트케이스를 제공한다.

본 실험에서는 스펙트럼 기반 기법으로 결함 위치를 식별할 수 없는 일부 버전을 제외한다. 다음의 프로그램들은 결함을 포함한 스테이트먼트를 모든 테스트케이스가 지나가지 않으므로 적용이 불가한 버전들이다. Printtokens(v4, v6), Printtokens2(v10), Replace(v12, v27, v32), Schedule(v5, v6, v9), Schedule2(v2, v9), Totinfo(v6, v10, v19, v21).

4.3 실험 결과

4.3.1 테스트케이스 그룹화 기법

65개의 유사도 및 거리 측정 기법에 테스트케이스 그룹화 기법에 적용하였다. 그림 2는 테스트케이스 그룹화 기법 적용 전, 후의 EXAM score의 증감률을 표현한 그래프이다. 총 65개의 기법에 적용한 결과 21개의 기법에서 향상된 결과가 나타났고 평균은 2.51%, 최대값은 47.51%, 최소값은 -22.48%, 분산은 12.71로 나타났다. 약 32.31%의 기법에서 성능향상이 되었지만, 나머지는 그대로이거나 성능이 저하된 일부 기법들도 있다.

테스트케이스 그룹화 기법의 특성상 의심도를 반복 계산, 누적을 하기 때문에 노이즈가 발생할 가능성을 예상했다. 그러므로 테스트케이스의 정보 중 테스트케이스

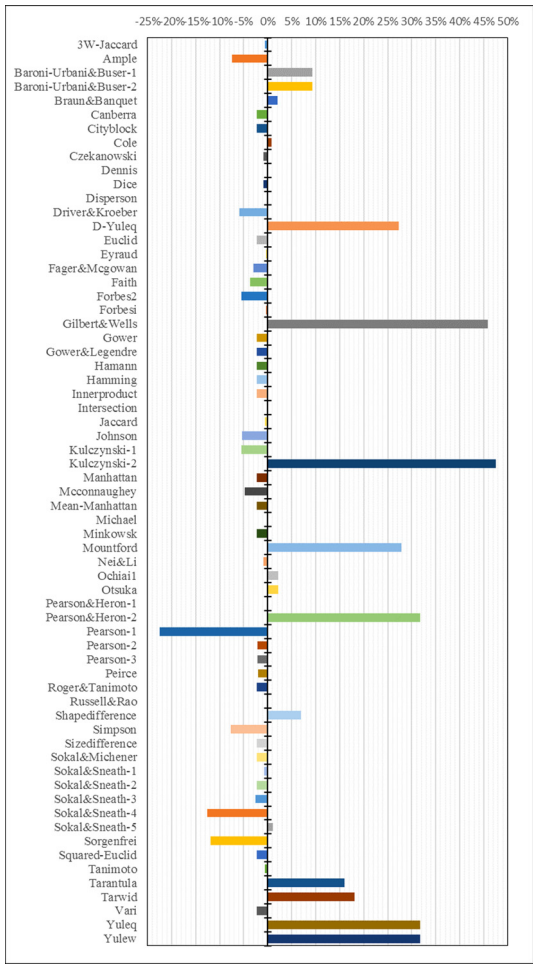

그림 2 테스트케이스 그룹화 기법의 적용 후의 EXAM score 증감률
Fig. 2 A rate of change of EXAM score before and after applying the test case grouping method

그룹화 기법의 성능 저하에 영향을 미치는 요소를 찾기 위해 상관분석을 실시한다.

독립변수는 실패 테스트케이스(FTC), 유일한 실패 테스트케이스 (Distinct FTC), 중복된 실패 테스트케이스 (Duplicated FTC), 성공 테스트케이스(PTC), 유일한 성공 테스트케이스 (Distinct PTC), 중복된 성공 테스트케이스 (Duplicated PTC)의 수이다. 종속변수는 버전별로 테스트케이스 그룹화기법을 적용하여 성능이 저하된 기법의 수로 설정한다. 6개의 독립 변수들 중, 통제가 가능한 중복된 실패, 성공 테스트케이스만을 대상으로 가설을 설정한다. 본 가설 검정의 목표는 모든 결함 버전에 65개의 기법을 적용했을 때, 성능 저하된 기법의 수에 영향을 미치는 테스트케이스의 요인을 찾기 위함이다.

가설검정1
$\begin{cases} H_0 : \rho_{FA} = 0 \\ H_a : \rho_{FA} \neq 0 \end{cases}$
F = 중복된 실패 테스트케이스의 수
A = 테스트케이스 그룹화 기법의 정확도

가설검정2
$\begin{cases} H_0 : \rho_{PA} = 0 \\ H_a : \rho_{PA} \neq 0 \end{cases}$
P = 중복된 성공 테스트케이스의 수
A = 테스트케이스 그룹화 기법의 정확도

상기의 가설1, 2의 검정을 위해 중복된 실패 테스트케이스와 중복된 성공 테스트케이스에 대하여 피어슨 상관 분석을 실시한다. 가설 검정은 고신뢰도를 위해 양측 검정으로 진행한다.

표 5에서 가설검정1의 결과, 중복된 실패 테스트케이스와 테스트케이스 그룹화기법의 정확도 간의 약한 음의 선형 관계(r=-0.145)가 나타났다. 그러나 p-value가 0.113으로 0.05보다 크므로 귀무가설을 기각할 수 없다. 즉, 중복된 실패 테스트케이스와 테스트케이스 그룹화 기법의 정확도 간에 상관관계가 있다고 할 수 없다. 가설검정2의 경우, 중복된 성공 테스트케이스와 테스트케이스 그룹화기법의 정확도 간에도 약한 음의 선형 관계(r=-0.242)가 나타났다. p-value도 0.007으로 0.01보다 작으므로 99% 신뢰수준으로 귀무가설을 기각할 수 있다. 즉, 중복된 성공 테스트케이스와 테스트케이스 그룹화 기법의 정확도 간에 상관관계가 있다고 할 수 있다.

4.3.2 테스트케이스 필터링 기법

테스트케이스 그룹화 기법은 의심도 계산의 노이즈를 누적하므로 성능이 저하된 기법도 일부 존재한다. 이런 한계점을 보완하고 정확도를 향상시키기 위해 테스트케이스 필터링 기법을 제안한다. 이 기법은 4.3.1장의 가설검정 결과를 기반으로 중복된 성공 테스트케이스를 제외하는 방법을 선택한다.

그림 3은 테스트케이스 그룹화 기법에 필터링 기법을 추가 적용한 후 기존 기법 대비 EXAM score 증감률을 나타낸 그림이다. 그림 1에 비해 전반적으로 정확성이 향상되었다. 총 65개의 기법에 중 54개의 기법에서 향상된 결과가 나타났고, 평균은 11.59%, 최대값은 51.31%, 최소값은 -10.36%, 분산은 11.93으로 나타났다. 적용 기법의 약 83.08%의 기법에서 성능향상이 되었다. 전체 359,826개의 테스트케이스를 105,959개로 줄여 효율성이 약 70.55% 향상되었다.

추가로 테스트케이스 필터 전략을 확장하여 실험을 수행하였다. 전략은 중복된 실패 테스트케이스와 중복된 실패, 성공한 모든 테스트케이스로 설정하여 실험을 진행하였다. 실험 결과는 표 6과 같다.

중복된 실패 테스트케이스를 필터링 한 경우, 정확성 측면에서, 19개의 기법에서 향상된 결과가 나타났다. 평균은 2.19%, 최대값은 48.37%, 최소값은 -23.52%, 분산은 13.42이며 적용 기법의 약 29.23%의 기법에서 성능향상이 되었다. 전체 359,826개의 테스트케이스를 354,474개로 줄여 효율성이 약 1.49% 향상 되었다. 중복된 실패, 성공 테스트케이스를 모두 필터링 한 경우, 정확성 측면에서, 59개의 기법에서 향상된 결과가 나타났다. 평균은 13.32%, 최대값은 51.24%, 최소값은 -10.22%, 분산은 11.70

표 5 테스트케이스 그룹화 기법의 정확도에 영향을 미치는 요인 분석

Table 5 Analysis of the affecting factors on accuracy of test case grouping method

Outperform	FTC	Distinct FTC	Duplicated FTC	PTC	Distinct PTC	Duplicated PTC
Pearson Correlation	-.462**	-.526**	-.145	-.334**	-.420**	-.242**
Significance (p-value)	.000	.000	.113	.000	.000	.007
Sum of Squares and Cross-products	54425.636	45254.182	9171.455	719229.909	444177.727	275052.182
Covariance	453.547	377.118	76.429	5993.583	3701.481	2292.102
N	121	121	121	121	121	121

**. Correlation is significant at the 0.01 level (2-tailed). *. Correlation is significant at the 0.05 level (2-tailed).

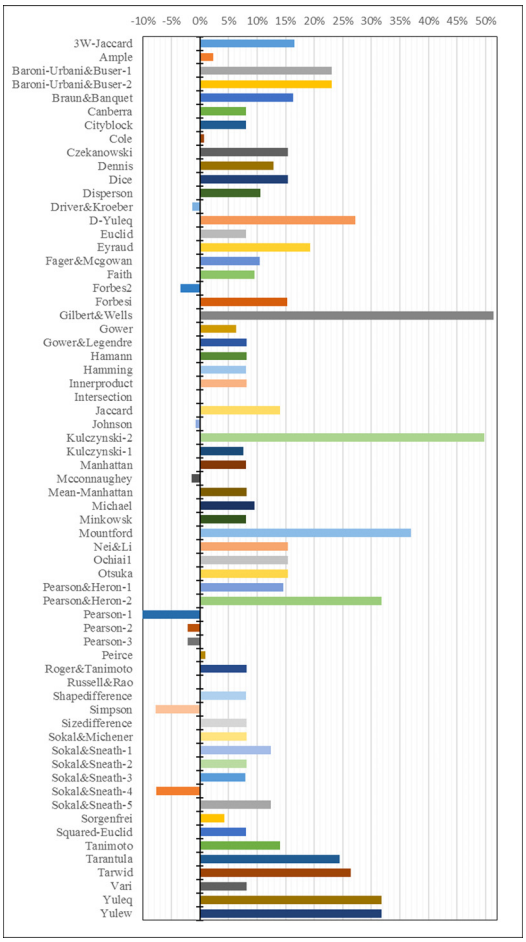

그림 3 테스트케이스 그룹화 및 필터링 기법 적용 전, 후의 EXAM score 증감률

Fig. 3 A rate of change of EXAM score before and after applying the test case filtering method

이며 적용 기법의 약 90.77%의 기법에서 성능향상이 되었다. 전체 359,826개의 테스트케이스를 100,607개로 줄여 효율성이 약 72.04% 향상 되었다.

4.3.1장의 가설검정1, 2의 신뢰도 향상을 위해 추가 실험을 진행했다. 테스트케이스 그룹화 기법 적용과 그룹화 및 필터링 기법을 모두 적용한 경우의 성능을 비교하였다. 필터링 전략은 중복된 성공 테스트케이스를 제거하는 방법을 사용한다. 55개의 기법에서 정확도가 향상 되었고, 평균은 9.16%, 최대값은 19.53%, 최소값은 0%, 분산은 5.32로 나타났다. 적용 기법의 약 84.62%의 기법에서 성능향상이 되었다. 성능 저하된 기법은 전혀 없고 전체 359,826개의 테스트케이스를 105,959개로 줄여 효율성이 약 70.55% 향상되었다. 표 7의 실험결과를 통해 테스트케이스 그룹화 기법에 중복된 성공 테스트케이스를 필터링하는 전략의 타당함을 입증한다.

최근 제안된 성능 우수 기법인 Naish1, Naish2, GP08, GP10, GP11, GP13, GP20, GP26, Heuristic1, Heuristic2, Heuristic3, Wong1, Wong2, Wong3, D star, SEM1, SEM2, SEM3은 적용대상에서 제외한다. 이 기법들은 실패, 성공 테스트케이스의 동시 적용 시 가장 좋은 성능을 낼 수 있도록 경험적으로 디자인되었기 때문에 테스트케이스 그룹화 기법의 장점을 반영할 수 없다. 실제 실험을 진행한 결과, 경험적 기법들에 테스트케이스 그룹화 및 필터링 기법을 적용하면 성능이 저하되었다.

표 8의 경험적으로 고안된 기법은 일반적 테스트케이스 구성 기법을 적용, 표 9의 65개의 유사도 및 거리 측정 기법들은 테스트케이스 그룹화 및 필터링 기법을 적용하였다. 필터링 전략으로는 유일한 성공 테스트케이스를 적용하여 전체 359,826개의 29.45%인 105,959개만 사용하였다.

표 6 테스트케이스 필터링 전략에 따른 성능 비교

Table 6 Performance comparison on test case filtering strategy

Filtering strategy		Test case grouping + filtering			
		Only grouping	Distinct PTC	Distinct FTC	Distinct F, PTC
Accuracy	Outperformed Average	2.51%	11.59%	2.19%	13.32%
	Outperformed Min	-22.48%	-10.36%	-23.52%	-10.22%
	Outperformed Max	47.51%	51.35%	48.37%	51.24%
	Outperformed Deviation	12.73	11.93	13.43	11.70
Efficiency	Selected test cases by filtering	359,826 /359,826	105,959 /359,826	354,474 /359,826	100,607 /359,826
	Efficiency of test case filtering	0%	70.55%	1.49%	72.04%
Number of Outperformed methods		21	54	19	59
Percentage of Outperformed methods		32.31%	83.08%	29.23%	90.77%
Number of Outperformed methods including no change		23	56	19	59
Percentage of Outperformed methods including no change		35.38%	86.15%	29.23%	90.77%

표 7 테스트케이스 그룹화 기법에 필터링 기법 추가 적용 전, 후의 성능 비교

Table 7 Performance comparison before and after additionally applying test case filtering method to grouping

		Test case grouping + filtering
Filtering strategy		Distinct PTC
Accuracy	Outperformed Average	9.16%
	Outperformed Min	0.00%
	Outperformed Max	19.53%
	Outperformed Deviation	5.32%
Efficiency	Selected test cases by filtering	105,959/359,826
	Efficiency of test case filtering	70.55%
Number of Outperformed methods		55
Percentage of Outperformed methods		84.62%
Number of Outperformed methods including no change		65
Percentage of Outperformed methods including no change		100.00%

표 8 스펙트럼 기반 경험적 기법의 EXAM score

Table 8 Exam score of heuristic method based on spectrum-based fault localization methods

Method		EXAM score	Method		EXAM score	Method		EXAM score
Naish	B	4.481	GP13	B	4.481	SEM2	B	14.929
	W	121.695		W	121.695		W	39.967
	M	63.088		M	63.088		M	27.448
GP08	B	12.197	GP20	B	4.527	SEM3	B	14.171
	W	32.08		W	117.227		W	39.209
	M	22.1385		M	60.877		M	26.69
GP10	B	4.236	GP26	B	15.975	Wong2	B	25.203
	W	95.178		W	32.034		W	41.288
	M	49.707		M	24.0045		M	33.2455
GP11	B	13.516	SEM1	B	16.119	Wong3	B	13.579
	W	33.399		W	36.186		W	29.638
	M	23.4575		M	26.1525		M	21.6085

B: Best case, W: Worst case, M: Median

EXAM score의 B(Best case)는 동일한 랭킹이 다수 존재할 때 가장 먼저 검사한 스테이트먼트에서 실제 결함이 발견되었을 경우를 의미한다. W(Worst case)는 B와 반대 개념으로 동일 랭킹 중 가장 마지막에 실제 결함을 찾은 경우를 의미한다. M(Median)은 Best와 Worst case의 중간 값을 의미한다.

Yuleq(4.365), Yulew(4.456), Intersection(4.713), Pearson-2(1.937), Pearson-3(1.937), Pearson&Heron-2(4.456), Russell&Rao(4.713), Simpson(4.713)에 테스트케이스 그룹화 및 필터링 기법을 적용 하는 것이 Naish(4.481),

표 9 테스트케이스 그룹화 및 필터링 기법 적용 후 EXAM score

Table 9 EXAM score after applying to test case grouping and filtering method

Method		EXAM score	Method		EXAM score	Method		EXAM score
3W-Jaccard	B	16.252	Gower	B	30.451	Pearson&Heron-1	B	15.249
	W	51.988		W	52.726		W	37.447
	M	34.12		M	41.5885		M	26.348
Ample	B	21.8	Gower&Legendre	B	30.839	Pearson&Heron-2	B	4.456
	W	46.004		W	48.329		W	115.671
	M	33.902		M	39.584		M	60.0635
BaroniUrbani&Buser-1	B	18.121	Hamann	B	30.839	Peirce	B	42.555
	W	53.857		W	48.329		W	462.015
	M	35.989		M	39.584		M	252.285
BaroniUrbani&Buser-2	B	18.121	Hamming	B	30.853	Roger&Tanimoto	B	30.839
	W	53.857		W	53.581		W	48.329
	M	35.989		M	42.217		M	39.584
Braun&Banquet	B	16.039	Innerproduct	B	30.839	Russell&Rao	B	4.713
	W	53.314		W	48.329		W	151.471
	M	34.6765		M	39.584		M	78.092
Canberra	B	30.853	Intersection	B	4.713	Shapedifference	B	34.242
	W	53.581		W	151.471		W	68.217
	M	42.217		M	78.092		M	51.2295
Cityblock	B	30.853	Jaccard	B	16.039	Simpson	B	4.713
	W	53.581		W	53.314		W	151.471
	M	42.217		M	34.6765		M	78.092
Cole	B	5.304	Johnson	B	14.531	Sizedifference	B	30.816
	W	260.552		W	50.884		W	48.306
	M	132.928		M	32.7075		M	39.561
Czekanowski	B	16.04	Kulczynski-2	B	14.487	Sokal&Michener	B	30.839
	W	52.036		W	51.761		W	48.329
	M	34.038		M	33.124		M	39.584
D-Yuleq	B	4.365	Kulczynski-1	B	19.053	Sokal&Sneath-1	B	16.316
	W	432.308		W	78.604		W	53.629
	M	218.3365		M	48.8285		M	34.9725
Dennis	B	15.875	Manhattan	B	30.853	Sokal&Sneath-2	B	30.839
	W	38.073		W	53.581		W	48.329
	M	26.974		M	42.217		M	39.584
Dice	B	16.04	Mcconnaughey	B	14.185	Sokal&Sneath-3	B	32.491
	W	52.036		W	40.641		W	63.968
	M	34.038		M	27.413		M	48.2295
Disperson	B	15.68	Mean-Manhattan	B	30.839	Sokal&Sneath-4	B	15.326
	W	45.9		W	48.329		W	49.439
	M	30.79		M	39.584		M	32.3825
Driver&Kroeber	B	14.487	Michael	B	16.126	Sokal&Sneath-5	B	17.101
	W	51.761		W	38.324		W	53.377
	M	33.124		M	27.225		M	35.239
Euclid	B	30.853	Minkowski	B	30.853	Sorgenfrei	B	16.039
	W	53.581		W	53.581		W	53.314
	M	42.217		M	42.217		M	34.6765
Yulew	B	4.456	Mountford	B	19.129	Squared-Euclid	B	30.853
	W	115.671		W	78.68		W	53.581
	M	60.0635		M	48.9045		M	42.217
Eyraud	B	14.273	Nei&Li	B	16.04	Tanimoto	B	16.039
	W	36.47		W	52.036		W	53.314
	M	25.3715		M	34.038		M	34.6765
Fager&Mcgowan	B	30.555	Ochiai	B	15.034	Tarantula	B	15.028
	W	58.905		W	51.058		W	50.764
	M	44.73		M	33.046		M	32.896
Faith	B	30.038	Otsuka	B	15.034	Tarwid	B	14.496
	W	47.972		W	51.058		W	40.442
	M	39.005		M	33.046		M	27.469
Forbes2	B	6.543	Pearson-1	B	26.327	Vari	B	30.839
	W	108.815		W	50.367		W	48.329
	M	57.679		M	38.347		M	39.584
Forbesi	B	16.179	Pearson-2	B	1.937	Yuleq	B	4.456
	W	52.837		W	249.894		W	115.671
	M	34.508		M	125.9155		M	60.0635
Gilbert&Wells	B	9.036	Pearson-3	B	1.937			
	W	81.276		W	249.894			
	M	45.156		M	125.9155			

B: Best case, W: Worst case, M: Median

GP10(4.236), GP13(4.481)에 비해 우수 또는 비슷한 정확도를 나타낸다. 또한, 전체 테스트케이스의 29.45%만 사용하므로 70.55% 효율성이 향상되었다.

5. 논의

Best case는 매우 낮지만 Worst case가 매우 높은 기법들이 일부 존재한다. 이는 실제 결함 스테이트먼트를 상위 랭킹에 두기는 하지만 동시에 다수의 동일한 랭킹을 갖는 스테이트먼트가 존재한다는 의미이다. 이러한 측면에서 성능 비교를 위해 EXAM score(M)을 기준으로 설정한다.

Wong3(21.6085), GP08(22.1385), GP11(23.4575)가 Eyraud(25.3715), Mcconnaughey(27.413), Michael (27.225), Pearson&Heron-1(26.348), Tarwid(27.469)에 테스트케이스 그룹화 및 필터링 기법을 적용한 것에 비해 약간 높은 정확성을 나타낸다.

EXAM score(M) 기준으로 성능 평가를 하면 기존 최우수 기법에 비해 정확도가 다소 낮다. 하지만 이는 전체 테스트케이스의 약 29.45%만 활용하여 도출된 결과이므로 효율성 측면에서 충분히 의미 있는 결과이다. 또한, 테스트 정확도는 필터링 전략에 따라 변하므로 개선된 필터링 전략을 세우면 정확성도 같이 향상 될 것으로 예상된다.

6. 결 론

스펙트럼 기반 결함위치식별 기법에서 실패 테스트케이스를 지나간 스테이트먼트에는 의심도를 부여하고 성공 테스트케이스를 지나간 스테이트먼트는 부여된 의심도 일부를 줄이는 역할을 한다. 그러므로 실패 테스트케이스의 역할이 매우 중요하며 실제로 실패 테스트케이스의 부재 시 결함의 위치를 찾을 수 없다. 하지만 이 기법에서 실패, 성공 테스트케이스를 동시에 입력하여 의심도를 계산하기때문에 실패 테스트케이스의 고유 특성을 반영할 수 없다는 한계점을 가지고 있다.

이러한 한계점을 보완하기 위해 본 논문에서 테스트케이스 그룹화 및 필터링 기법을 제안한다. 또한, 이 기법의 유효성 평가를 위해 65개의 유사도 및 거리 측정 기법에 적용하여 성능을 비교한다.

테스트케이스 그룹화 및 필터링 기법(필터링 전략: 유일한 성공 테스트케이스)을 적용한 결과, 기존 기법 대비 평균 11.59%의 정확성이 향상되었다. 전체 65개의 기법 중 83.08%인 54개의 기법에서 성능이 향상 되었으며 테스트케이스는 전체 359,826개의 29.45%인 105,959개로 줄여 효율성이 약 70.55% 향상되었다.

상기 실험 결과를 기반으로 새로운 스펙트럼 기반 기법 제안을 하는 것 보다 테스트케이스 그룹화 및 필터링 기법의 적용만으로 더 높은 테스트 성능 향상이 가능하다. 즉, 이 기법의 적용만으로 기존 최우수 기법들보다 정확하고 효율적인 테스트가 가능하다.

7. 향후 연구

테스트케이스 필터링 전략의 추가로 테스트의 정확성과 효율성을 보다 향상시킬 수 있다. 본 연구팀에서는 실패 테스트케이스의 고유 특성을 부각시킬 수 있는 일부 성공 테스트케이스를 선택하는 연구를 진행하고 있다. 또한 실험 대상의 확장과 다양성을 위해 space, gzip, grep 등의 프로그램에 적용 준비 중이다.

References

[1] David Landsberg, Hana Chockler, Daniel Kroening, and Matt Lewis, "Evaluation of Measures for Statistical Fault Localisation and an Optimising Scheme," *Fundamental Approaches to Software Engineering*, Springer Berlin Heidelberg, pp. 115-129, 2015.

[2] Costa, Pedro, João Gabriel Silva, and Henrique Madeira, "Practical and representative faultloads for large-scale software systems," *Journal of Systems and Software 103*, pp. 182-197, 2015.

[3] Noor, Tanzeem, and Hadi Hemmati, "Test case analytics: Mining test case traces to improve risk-driven testing," *Software Analytics (SWAN) IEEE 1st International Workshop on. IEEE*, pp. 13-16, 2015.

[4] Jeongho Kim, Jonghee Park and Eunseok Lee, "A New Spectrum-based Fault Localization with the Technique of Test Case Optimization," *Journal of Information Science and Engineering (JISE)*, Vol. 32, No. 1 Jan, 2016, in press.

[5] Abreu, Rui, Peter Zoeteweij, and Arjan JC Van Gemund, "On the accuracy of spectrum-based fault localization," *Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION (TAICPART-MUTATION), IEEE*, pp. 89-98, 2007.

[6] Xie, Xiaoyuan, W. Eric Wong, Tsong Yueh Chen, and Baowen Xu, "Metamorphic slice: An application in spectrum-based fault localization," *Information and Software Technology 55.5*, pp. 866-879, 2013.

[7] Wong, W. Eric, and Yu Qi, "Effective program debugging based on execution slices and inter-block data dependency," *Journal of Systems and Software 79.7*, pp. 891-903, 2006.

[8] Wong, W. Eric, Debroy, V., Ruizhi Gao, and Yihao Li, "The dstar method for effective software fault localization," *Reliability, IEEE Transactions on 63.1*, pp. 290-308, 2014.

[9] Wong, W. Eric, Vidroha Debroy, and Byoungju Choi, "A family of code coverage-based heuristics

for effective fault localization," *Journal of Systems and Software 83.2*, pp. 188-208, 2010.
[10] Naish, Lee, Hua Jie Lee, and Kotagiri Ramamohanarao, "A model for spectra-based software diagnosis," *ACM Transactions on software engineering and methodology (TOSEM) 20.3*, Article 11, 2011.
[11] Xie, Xiaoyuan, Tsong Yueh Chen, Fei-Ching Kuo, and Baowen Xu, "A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization," *ACM Transactions on Software Engineering and Methodology (TOSEM) 22.4*, Article 31, 2013.
[12] Kim, Jeongho, Jonghee Park, and Eunseok Lee, "A new hybrid algorithm for software fault localization," *Proc. of the 9th International Conference on Ubiquitous Information Management and Communication. ACM*, 2015.
[13] Kim, Jeongho, and Eunseok Lee, "Empirical evaluation of existing algorithms of spectrum based fault localization," *Information Networking (ICOIN), 2014 International Conference on. IEEE*, pp. 346-351, 2014.
[14] Misherghi, Ghassan, and Zhendong Su, "HDD: hierarchical delta debugging," *Proc. of the 28th international conference on Software engineering*, pp. 142-151, 2006.
[15] Groce, Alex, Mohammad Amin Alipour, Chaoqiang Zhang, Yang Chen, and John Regehr, "Cause reduction: delta debugging, even without bugs," *Software Testing, Verification and Reliability* 26, No. 1 pp. 40-68, 2016.
[16] Jia, Yue, and Mark Harman, "An analysis and survey of the development of mutation testing," *Software Engineering, IEEE Transactions on 37.5*, pp. 649-678, 2011.
[17] Debroy, Vidroha, and W. Eric Wong, "Combining mutation and fault localization for automated program debugging," *Journal of Systems and Software 90*, pp. 45-60, 2014.
[18] Papadakis, Mike, and Yves Le Traon, "Metallaxis-FL: mutation-based fault localization," *Software Testing, Verification and Reliability*, Vol. 25, No. 5-7, pp. 605-628, 2013.
[19] Gopinath, Divya, Razieh Nokhbeh Zaeem, and Sarfraz Khurshid, "Improving the effectiveness of spectra-based fault localization using specifications," *Automated Software Engineering (ASE), Proceedings of the 27th IEEE/ACM International Conference on. IEEE*, pp. 40-49, 2012.
[20] Bekkouche, Mohammed, Hélene Collavizza, and Michel Rueher, "LocFaults: A new flowdriven and constraint-based error localization approach," *Symposium on Applied Computing (SAC)*, ACM., Vol. 15, 2015.
[21] Khoshnood, Sepideh, Markus Kusano, and Chao Wang, "Concbugassist: Constraint solving for diagnosis and repair of concurrency bugs," *International Symposium on Software Testing and Analysis*, pp. 165-176, 2015.
[22] Ke, Yalin, Kathryn T. Stolee, Claire Le Goues and Yuriy Brun, "Repairing programs with semantic code search," *Proc. of the 30th IEEE/ACM International Conference On Automated Software Engineering (ASE)*, pp. 295-306, 2015.
[23] Yoo, Shin, "Evolving human competitive spectra-based fault localisation techniques," *Search Based Software Engineering*, Springer Berlin Heidelberg, pp. 244-258, 2012.
[24] Yoo, Shin, and Mark Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability* 22.2, pp. 67-120, 2012.
[25] Daniel, Patrick, "On improving the accuracy of spectrum-based fault localization," Doctoral dissertation, Swinburne University of Technology, 2014.
[26] Do, Hyunsook, Sebastian Elbaum, and Gregg Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," *Empirical Software Engineering*, 10.4, pp. 405-435 2005.
[27] Bayraktar, "ELA: An automated statistical fault localization technique," Doctoral dissertation, Middle East Technical University, 2015.
[28] Software-artifact Infrastructure Repository, [Online]. Available: http://sir.unl.edu/portal/index.php

김 정 호
2012년 한성대학교 산업시스템공학과 학사. 2013년 현재 성균관대학교 전자전기컴퓨터공학부 석박사통합과정. 관심분야는 결함위치식별, 자동 디버깅, 소프트웨어 테스팅

이 은 석
정보과학회논문지
제 43 권 제 7 호 참조