소프트웨어의 선택적 교호작용 테스팅

고병각°, 이상용, 장중순, 최경희*, 박승규*, 정기현**
아주대학교 산업공학과, 컴퓨터공학과*, 전자공학과**
{rockliky°, biztech, jsjang, khchoi, sparky, khchung }@ajou.ac.kr

Selective interaction testing for software

Byunggak Ko° , Sanyong Lee, Jungsoon Jang, Kyunghee Choi*, Seung-Kyu Park* Kihyun Chung**
Industrial Engineering, Computer Engineering*, Electronic Engineering** department
Ajou University

요 약

소프트웨어 테스팅에서 테스트 스위트(suite)의 수를 줄이면서도 테스트 커버리지나 오류검출에 있어서 효과적인 방법을 찾기 위한 많은 연구가 시도되어 왔다. 안정성이 높은 설계가 되도록 설계 조건을 결정하기 위하여 제어인자들의 직교 배열을 사용하는 시험방법인 교호작용 테스팅 기법이 소프트웨어의 테스트에서도 효과가 매우 높다는 것이 실험으로 증명되고 있다. 소프트웨어는 상대적으로 몇 안 되는 조건들의 조합들로 오류가 발생할 가능성이 높다는 특징을 가지고 있다. 따라서, 파라미터 간 교호작용 강도 t를 갖는 t-way 테스팅을 통해 효과적으로 테스트 스위트를 줄이면서 많은 오류를 검출할 수 있다. 그러나 t값을 증가시키면 테스트 스위트의 수가 늘어난다는 단점이 발생한다. 또한, 어떠한 파라미터들이 서로 교호작용을 일으키는 것인지 알 수 없는 상황을 가정한 단순한 교호작용 테스팅은 자칫 서로 관련 없는 파라미터들도 교호작용 테스팅에 참여하기 때문에 테스트의 낭비가 있을 수 있다는 단점이 있다. 이에, 본 논문에서는 소프트웨어의 입력과 출력간의 관계를 바탕으로 시스템 I-O관계도를 작성한 후, 이를 바탕으로 각 출력에 대한 교호작용을 일으킬 수 있는 파라미터를 중심으로 테스트 스위트를 생성하는 Selective Covering Array를 제안한다.

1. 서 론

교호작용(interaction)에 대한 영향은 주로 농업, 화학, 품질관리 분야 등에서 많은 연구가 진행되어 왔다. 최근에는 소프트웨어 테스팅 분야에서도 효과가 있음이 밝혀지면서 그 적용방법에 대한 연구가 활발히 진행되고 있다. Kuhn, Wallace, Gallo[9], Nair [11], Wallace와 kuhn[12], Kuhn과 Reilly[13] 등은 기 개발된 소프트웨어 오류 데이터를 분석한 후, 많은 오류가 2개 이상의 파라미터간 교호작용에 의해 파생되는 것을 밝혀 내면서, 교호작용 테스팅의 당위성을 입증하였다. 교호작용 테스트 방법은 테스트 스위트(suite)의 수를 많이 줄일 수 있는 장점이 있어, 테스트에 드는 시간과 예산의 제약으로 인해 모든 경우의 수를 철저히 테스트 할 수 없을 경우 특히 효율적인 테스트 방법이다. 교호작용 테스팅은 오류가 일어날 가능성이 높은 것에 우선순위를 두고 테스트 하는 위험 기반 테스팅(risk-based testing) 개념의 일종으로 생각할 수도 있다. 현재 교호작용 테스팅을 위한 알고리즘 및 프레임워크에 대한 연구[3,4,8,10,14]가 기존의 조합최적화 분야의 지식을 응용하여 진행되고 있다.

그러나, 이러한 연구들은 대부분 교호작용이 몇 가지 파라미터에 의해서 생기는 지에 대한 정보가 없기 때문에 pairwise가 충분한지 아니면 3-way testing이 필요한지 등 적절한 coverage를 제시할 수 없다는 단점이 있다. 본 논문에서는 이러한 문제의 대안으로 교호작용을 일으키는 파라미터를 명세서를 토대로 분석하여 이를 기반으로 어떤 파라미터들이 어떤 결과에 영향을 미치는 지 알아내어 이것들을 효과적으로 테스트 스위트를 생성하는 방안을 제시한다.

2. 교호작용 테스팅

교호작용 테스팅이란 몇 가지 특정한 조건들이 같이 발생될 때 소프트웨어 오류가 일어 날 가능성이 높다는 점에서 착안하여 시스템에 입력이 되는 파라미터들간의 교호작용을 테스트 하는 것이다. 이 테스트는 R. Brownlie가 AT&T PMX/StarMail를 테스트하면서 직교배열표를 이용한 것과[1] David M. Cohen 등이 Bellcore의 AETG 시스템 테스트를 n-way 파라미터 조합을 통해 실시함으로써 테스트 케이스를 줄이면서 좋은 커버리지를 가지는 실험[2] 에서 보듯이 우수한 테스팅 방법으로 입증되었다.

AETG 알고리즘은 상업용 소프트웨어로 특허를 받았기 때문에 그 소스가 공개되지 않았다. Yu Lei 등은 IPO(In-Parameter-Order) 알고리즘을 구현하여 all-pairwise 테스팅을 위한 테스트 케이스를 생성하였다. Williams 등은 네트워크 노드 간의 교호작용 테스팅 실험을 실시하였다[4]. Dalal 등은 all-pairwise 교호작용 테스팅을 통해서 실제로 소프트웨어 시스템에서 수많은 오류를 검출 하였다[5]. Burr 등은 테스트가 커버리지 면에서도 효과적임을 밝혔다[6]. IPO 알고리즘은 모든 파라미터에 대한 pairwise(2-way) 테스트 케이스만 생성할 수 있다는 단점이 있다.

Myra B. Cohen 등은 시뮬레이티드 어닐링 알고리즘을 이용한 유연하고 확장 가능한 n-way 교호작용 테스팅을 소개하였다[7][8]. 이 테스팅 방법은 파라미터들이 서로 다른 개수의 값을 가지고 있을 경우와 모든 파라미터에 대해서 t-way조합을 실시하는 동시에 몇 가지 파라미터에 대해서는 더욱 강한 강도의 t' -way 조합을 지원한다. 예를 들어, 총 11개의 파라미터 A~K가 다음 [표1]와 같은 값들을 가지고 있다고 가정하자.

[표 1] 파라미터의 값

A	B	C	D	E	F	G	H	I	J	K
0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2		

모든 파라미터의 가능한 모든 조합을 입력하여 소프트웨어를 테스트 하려면 3^9*2^2 =78,732 개의 테스트 케이스가 필요하다. 하지만 11개의 파라미터에 대해서 pairwise테스팅을 하는 동시에 A~C, D~F, G~J에 대해서는 3-way 테스팅을 위하여 Cohen의 시뮬레이티드 어닐링 알고리즘을 사용하여 테스트 케이스를 생성하면 27개의 테스트 케이스만이 생성된다. 이는 모든 가능한 조합의 수인 78,732개의 0.034%에 불과한 숫자이다. 이를 기호를 사용하여 표시하면

$$VCA(27:2,3^9,2^2,CA(27:3,3^3)^3)$$

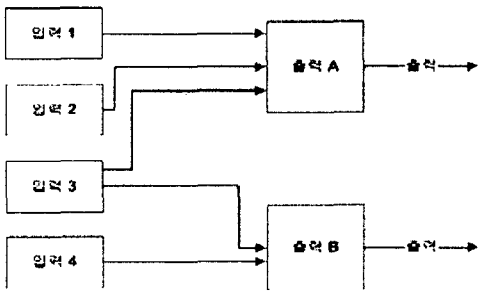
과 같이 나타낼 수 있다[7].

Variable strength interaction testing을 variable strength covering array(VCA)로 표현 할 때 파라미터는 왼쪽에서 오른쪽으로 나열되기 때문에 각 열의 순서는 중요하다. 즉, 파라미터는 왼쪽에서 오른쪽으로 적절히 배치되어야 한다. 또한 모든 파라미터에 대해서 기본적으로 pairwise, 3-way를 만들어야 한다. 이 두 가지는 variable strength interaction testing의 제약이 될 수 있다.

3. 선택적 교호작용 테스팅

본 논문에서는 입력과 출력의 관계를 규명함으로써 교호작용 테스팅 기법의 단점을 해결하고자 하였다. 여러 가지 입력과 이에 상응하는 여러 가지 출력을 가지는 시스템(특히 임베디드 시스템)의 입력과 출력 관계를 분석하여 하나의 출력에 관계된 입력들을 full-way(factorial) 조합을 하여 적어도 이런 조합들이 테스트 스위트에 하나씩 생성하도록 하면 효과적인 테스트가 가능하다.

[그림 1]의 시스템 I-O(Input-Output) 관계도를 보면 이 시스템의 입력은 4개이고 이 입력 값들은 출력 A와 출력 B 의 연산을 위한 값들이다. 입력 1, 입력 2, 입력 3은 출력 A에 들어가는 값이고, 입력 3, 입력 4는 출력 B에 들어가는 값이다. 즉 이 시스템은 각 입력은 4개이지만 이 입력 값들을 처리하는 내부 출력은 각각 필요로 하는 입력 값들이 다르다.



[그림 1] 예) 시스템 I-O 관계도

출력 A의 경우, 입력 1, 입력 2, 입력 3과 관련이 있다. 즉 출력 A를 테스트 하기 위해서는 입력 1, 입력 2, 입력 3 이 가지는 값의 모든 경우를 조합하여 테스트를 하면 커버리지가 100% 만족 할 수 있을 것이다. 또한, 출력 B는 입력 3 과 입력 4가 가지는 값을 모두 조합하면 마찬가지로 커버리지 100%를 만족할 수 있을 것이다. 즉 입력 1과 입력 4, 입력2 와 입력 4는 서로 교호작용이 일어날 수 없으므로 테스트에서 제외할 수도 있다. 여기서 각 입력의 값을 어떻게 선정할 것인가가 문제가 될 수 있는데 보통 동치분할법, 경계값 분석 등으로 할 수 있다.

각 입력이 가지는 값이 [표 3]과 같다고 가정하자. 시스템의 입력값에 대한 커버리지 100%를 만족하기 위하여는 다음과 같이 테스트 할 수 있다. 모든 경우의 수를 테스트 하기 위해서는 테스트 스위트는 2*2*3*2=24개가 될 것이다. 하지만 위에서 분석한 입력, 출력 관계에 따라 작성된 테스트 스위트는 출력 A를 테스트하기 위해서는 2*2*3=12가지 테스트 케이스가 필요하고, 출력 B를 테스트 하기 위해서는 3*2=6가지가 필요하여 총 18개가 될 것이다. 그러나 본 논문에서 제안하는 selective covering array(SCA) 알고리즘을 이용하여 작성된 테스트 스위트는 12개로 구성될 수 있다.

[표 2] 각 입력이 가지는 값

입력 1	입력 2	입력 3	입력 4
0	0	0	0
1	1	1	1
		2	

SCA 가 covering array 와 다른 점은 변수들간의 교호작용 테스팅을 선택적으로 할 수 있다는 점이다. SCA 는 기본적으로

$$SCA(N;t(v_1^{p1},\cdots,v_k^{pk})),\ 1\le t,\ 1\le k$$

의 형태로 표현하며 그 의미는 다음과 같다.

- N : 행의 크기(size) , 즉 테스트 케이스 수

- t : 강도(strength), 즉 t-way 교호작용 강도
- p_k: 파라미터
- v_k : 각 파라미터 p_k 가 가지는 값의 개수
- k : 파라미터의 개수

예를 들면 v_1^{p1} 은 p_1 파라미터가 v_1 개의 값을 가지고 있다는 것을 의미한다.

SCA 는

$$SCA(N;t(v1^{p1},v2^{p2},v3^{p3}), u(v1^{p2},v2^{p4}))$$

등으로 교호작용 강도를 확장하여 표현할 수도 있다. 예를 들면, p_1, p_2, p_3 파라미터는 t-way 로 조합하며 p_2, p_4 파라미터는 u-way 로 조합한다. t, u 는 시스템 I-O 관계도로 분석될 수 있다. [그림 1]을 예로 들어 SCA 를 설명하면 다음과 같다. 입력 파라미터인 입력 1 을 A, 입력 2 를 B, 입력 3 을 C, 입력 4 를 D 라고 하고 각 파라미터가 가지는 값이 [표 3]과 같다고 할 때, 출력 A 와 출력 B 를 각각 100% 교호작용 커버리지를 만족하는 모든 테스트의 경우는 다음과 같다.

- 출력 A : 입력 A,B,C 의 3-way 조합
- 출력 B : 입력 C,D 의 2-way(pairwise) 조합

이것은 $SCA(12;3(2^A,2^B,3^C),2(2^C,2^D))$로 표현할 수 있으며 생성된 테스트 스위트는 [표 4]와 같다.

[표 3] $SCA(12;3(2^A,2^B,3^C),2(2^C,2^D))$

No	A	B	C	D
1	0	0	0	0
2	0	0	1	1
3	0	0	2	0
4	0	1	0	0
5	0	1	1	1
6	0	1	2	1
7	1	0	0	1
8	1	0	1	0
9	1	0	2	0
10	1	1	0	0
11	1	1	1	1
12	1	1	2	1

4. 결론

본 논문에서는 선택적 교호작용 테스팅(selective interaction testing)을 위하여 selective covering array(SCA)를 제안하였다. 이 테스트 스위트 생성 방법은 기존의 n-way 테스팅 방법 및 covering array 와 다음과 같은 점에서 차이점이 있다.

1. SCA 는 입력값 들이 출력에 영향을 토대로 시스템 I-O 관계도를 작성한 후, 이를 토대로 각 출력의 100% 커버리지를 만족하는 테스트 케이스를 생성한다.
2. SCA 는 VCA 처럼 왼쪽에서 오른쪽으로 파라미터 순서를 배열하는 제약이 없다.
3. SCA 는 변수들간의 교호작용 테스팅을 출력별로 선택적으로 할 수 있도록 테스트 케이스를 조합할 수 있다.

5. References

[1] R. Brownlie, J. Prowse, and M.S. Phadke, "Robust Testing of AT&T PMX/StarMail Using OATS," AT&T Technical J., vol. 71, no. 3, pp. 41-47, May/June 1992.

[2] D.M. Cohen, S.R. Dalal, J. Parelius, and G.C. Patton, "The Combinatorial Approach to Automatic Test Generation," IEEE Software, vol. 13, no. 5, pp. 83-88, Sept. 1996.

[3] Yu Lei; Tai, K.C., "In-parameter-order: a test generation strategy for pairwise testing," High-Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International 13-14 Nov. , pp. 254 – 261,1998.

[4] A. W. Williams., "Determination of test configurations for pair-wise interaction coverage," In Proc. Thirteenth Int. Conf. Testing Communication Systems, pp. 57–74, 2000.

[5] S. R. Dalal, A. J. N. Karunanithi, J. M. L. Leaton, G. C. P. Patton, and B. M. Horowitz. "Model-based testing in practice," In Proc. of the Intl. Conf. on Software Engineering(ICSE '99), New York, pp. 285-94, 1999.

[6] K. Burr and W. Young, "Combinatorial test techniques : Table-based automation, test generation and code coverage," In Proc. of the Intl. Conf. on Software Testing Analysis & Review, San Diego, 1998.

[7] M. B. Cohen, C. J. Colbourn, J.S. Collofello, P. B. Gibbons and W. B. Mugridge, "Variable Strength Interaction Testing of Components," In Proc. of the Intl. Computer Software and Applications Conference, (COMPSAC 2003), Dallas TX, 2003.

[8] M. B. Cohen, C. J. Colbourn and A.C.H. Ling, "Augmenting simulated annealing to build interaction test suites," 14th IEEE Intl. Symp. on Software Reliability Engineering(ISSRE 2003), Denver CO, pp. 394-405, November 2003.

[9] D. Richard Kuhn, Dolores R. Wallace, Albert M. Gallo Jr. , "Software Fault Interactions and Implications for Software Testing," IEEE transactions on Software Engineering, vol. 30, no. 6, June 2004.

[10] Ren'ee C. Bryce , Charles J. Colbourn, Myra B. Cohen, "A Framework of Greedy Methods for Constructing Interaction Test Suites," ICSE'05, St. Louis, Missouri, May 15–21, 2005.

[11] V.N. Nair, D.A. James, W.K. Erlich, and J. Zevallos, "A Statistical Assessment of Some Software Testing Strategies and Application of Experimental Design Techniques," Statistica Sinica, vol. 8, no. 1, pp. 165- 184, 1998.

[12] D.R. Wallace and D.R. Kuhn, "Failure Modes in Medical Device Software: An Analysis of 15 Years of Recall Data," Int'l J. Reliability, Quality and Safety Eng., vol. 8, no. 4, 2001.

[13] D.R. Kuhn and M.J. Reilly, "An Investigation of the Applicability of Design of Experiments to Software Testing," Proc. 27th NASA/IEEE Software Eng. Workshop, Dec. 2002.