2진수를 활용한 MCC 테스트 케이스 생성기 설계 및 구현

(A Design and Implementation of MCC Test Cases Generator using Binary Numbers)

이로빈*, 남영호*
(Robin Lee, Young Ho Nam)

요 약

소프트웨어 테스트는 소프트웨어 개발 프로세스의 필수적인 부분이다. 특히 MC/DC(Modified Condition / Decision Coverage)는 복잡한 조건과 결정 구조를 효과적으로 검증하여 소프트웨어의 안정성과 신뢰성을 높이는 데 사용된다. 본 연구에서는 MC/DC 수행의 최대 커버리지 값 확인을 위해 2진수를 활용한 MCC(Multiple Condition Coverage) 테스트 케이스 생성하는 MTC(MCC Test Cases) 생성기를 제안한다. 제안한 MTC 생성기는 TCAS(Traffic alert and Collision Avoidance System)-II 명세서 조건의 일부를 사용하여 CSV(Comma-Separated Values) 파일로 변환 후, VectorCAST 프로그램을 통해 커버리지 결과를 확인하였다. 그 결과 MCC 테스트 케이스로 MC/DC를 수행하여 TCAS-II 명세서의 각 조건들의 MC/DC 수행 시 최대 커버리지 값을 확인하였다. 이는 MC/DC 수행 시 커버리지 최댓값을 확인 함으로써 MC/DC 테스트 케이스 검증 연구에 도움이 된다. 또한 더 많은 테스트 케이스를 통해 결함발견 가능성도 증가시킨다. 이를 통해 소프트웨어 테스트의 커버리지 검증의 효율성과 소프트웨어의 품질 및 안정성을 향상하는 데 기여할 수 있다.

■ 중심어 : 소프트웨어 테스트 ; 테스트 케이스 ; 커버리지 ; 다중조건 커버리지 ; 변경조건 결정 커버리지

Abstract

Software testing is essential in the software development process. Modified Condition / Decision Coverage (MC/DC) is a test case derivation technique that enhances the stability and reliability of software by effectively verifying complex conditions and decision structures. We propose the MCC Test Cases (MTC) generator in this study. This generator generates Multiple Condition Coverage (MCC) test cases using binary numbers to confirm the maximum coverage value of MC/DC testing. The proposed MTC generator utilizes some conditions from the Traffic Alert and Collision Avoidance System (TCAS)-II specification. It converts them into a Comma-Separated Values (CSV) file and then validates the coverage results through the VectorCAST program. So, MC/DC testing was performed using the MCC test case to confirm the maximum coverage value when performing MC/DC tests for each condition of the TCAS-II specification. This research is helpful for the verification of MC/DC test cases by confirming the maximum coverage value when performing MC/DC tests. Moreover, having more test cases increases the likelihood of discovering defects. Therefore, it can improve the efficiency of software test coverage verification, as well as the quality and stability of software.

■ keywords : Software Test ; Test cases ; Coverage ; MCC ; MC/DC

Ⅰ. 서 론

소프트웨어 테스트는 정적과 동적 테스트로 구분된다. 정적 테스트는 코드를 실행하지 않고 분석하는 방법이며, 동적 테스트는 실제 코드를

* 정회원, 경상국립대학교 컴퓨터공학과
이 연구는 2024년도 경상국립대학교 발전기금재단 재원으로 수행되었음
접수일자 : 2024년 05월 13일
수정일자 : 2024년 06월 03일
게재확정일 : 2024년 06월 25일
교신저자 : 남영호 e-mail : yhnam@gnu.ac.kr

실행하여 테스트하는 방법이다. 동적 테스트는 블랙박스 테스트와 화이트박스 테스트로 분류된다. 블랙박스 테스트는 소프트웨어의 내부 구조나 작동 원리를 모르는 상태에서 입력과 출력만을 통해 기능을 테스트하는 방법이다. 화이트박스 테스트는 소프트웨어의 내부 구조와 작동 원리를 분석하여 테스트하는 방법이다. 화이트박스 테스트는 내부 소스코드를 테스트하기 위해 코드 커버리지를 수행하게 된다. 코드 커버리지는 해당 코드가 얼마나 충족되었는지를 나타내는 지표 중 하나이고 퍼센트화(%)하여 수치상으로 표시한다. 코드 커버리지 종류는 <그림 1>과 같다.

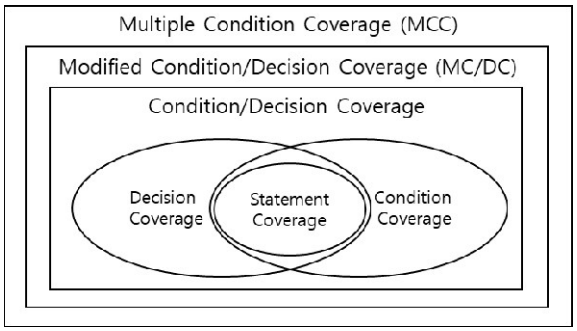

그림 1. 코드 커버리지

MCC(Multiple Condition Coverage)는 분기점에서 가능한 모든 조건의 논리적 조합을 테스트하는 방법이다. MCC 테스트 케이스 수는 조건문의 개수 N에 대해 2^N 개가 나오므로 가장 넓은 커버리지이다.

MC/DC(Modified Condition / Decision Coverage)는 각 개별 조건식이 다른 조건식에 영향을 주지 않고 전체 조건식의 결과에 독립적으로 영향을 주는 테스트를 말한다. MC/DC 테스트 케이스 수는 조건문의 개수 N에 대해 최소한 N+1개 이상이 나온다[1-7].

이러한 테스트와 코드 커버리지 기법은 소프트웨어의 복잡한 조건과 결정 구조를 효과적으로 검증하여, 소프트웨어의 안정성과 신뢰성을 높이는 데 중요한 역할을 한다[8-11]. 특히, 임베디드 소프트웨어와 같이 안정성이 중요한 분야에서 효율적인 테스트 기법은 꼭 필요한 연구 분야이다[12-16].

본 논문은 주어진 소스코드의 MC/DC 테스트 케이스의 최대 커버리지 값 확인을 위한 MCC 테스트 케이스 생성에 관한 연구이다. MCC 테스트 케이스 생성을 위해 TCAS-II[17-19]의 명세서에 기술된 일부 조건식들을 사용하였다. 각 조건식에 대한 테스트 케이스들은 범용적으로 사용이 가능한 CSV(Comma-Separated Values) 파일로 변환된 후, MC/DC 테스트를 수행하는 데 활용된다. 검증은 상용 커버리지 측정 소프트웨어 도구인 VectorCAST(2024) 프로그램을 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 소프트웨어 테스트 케이스에 대한 내용을 정의하고 프로그램 설계 시에 고려 사항들을 기술한다. 3장에서는 MCC 테스트 케이스를 생성을 위한 MTC 생성기(MCC Test Cases Generator) 설계에 관하여 설명한다.

4장에서는 MTC 생성기의 구현에 관하여 기술한다. 5장에서는 본 연구의 결과에 관해 검증하며, 마지막 6장에서는 결론을 기술한다.

Ⅱ. 소프트웨어 테스트 케이스

소프트웨어 테스트 케이스는 특정한 프로그램 경로를 실행하여 입력값, 실행 조건, 예상 결과의 집합을 확인하는 것이다. 이를 통해 요구사항이 제대로 구현되었는지 검증하고 잠재적인 결함을 최소화할 수 있다.

본 연구에서는 조건에 따른 모든 논리적 조합인 MTC를 생성하기 위해 프로그래밍 언어 중 하나인 C언어를 선정하고 <표 1>과 같은 TCAS-II 명세서에 기술된 일부 조건식을 사용하였다.

표 1. TCAS-II 명세서의 조건식

	Specifications
1	a && (!b || !c) && d || e
2	a && c && (d || e) && h || a && (d || e) && !h || b && (e || f)
3	(a && c || b && d) && e && (f || (i && (g && j || h && k)))
4	(a && c || b && d) && e && (f && g || !f && h)
5	!e && f && !g && !a && (b && c || !b && d)

Ⅲ. MTC 생성기 설계

1. 프로그램 설계

본 연구에서 개발하는 MCC 테스트 케이스 생성 프로그램명은 “MTC Generator”이다. GUI 모듈은 소프트웨어 사용자와 인터페이스를 제공하며, Processing 모듈은 C언어파일 내부의 조건문 파싱 및 CSV 파일을 생성한다.

2. 파싱 및 CSV 생성 기능 설계

파싱은 <그림 2>의 C언어 파일 내부 조건문을 추출하는 기능을 제공한다. 먼저, C언어 파일을 로드하여 if문이 포함된 개별 조건들을 인식한다. 인식된 개별 조건문의 개수 N에 대해 2^N 개로 최대 논리적 조합인 MTC를 생성한다. 다음으로 생성된 MTC를 CSV 형식의 파일로 저장한다.

```c
#include<stdio.h>
#include<stdbool.h>

bool a, b, c, d, e;

int main(void) {
    if ( a && (!b || !c) && d || e ) {
        printf("이하 코드 생략₩n");
    }
    else {
        printf("End");
    }
    return 0;
}
```

그림 2. C언어 파일

3. 메인화면 구성

<그림 3>은 MTC 생성기의 UI를 설계한 것이며, 각 필드와 버튼의 기능은 다음과 같다.

- (.c)Path: 소프트웨어 코드인 C언어 파일을 불러오는 버튼
- Target: MTC가 저장될 CSV 파일을 저장하기 위한 경로
- Generator: MTC 파일 생성 버튼
- 파일생성 진행률을 표시하는 진행바
- 파일생성 중 오류 및 생성정보를 나타내는 출력란

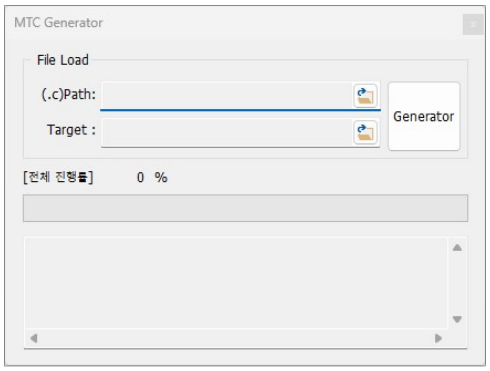

그림 3. MTC 생성기 UI

Ⅳ. MTC 생성기 구현

1. 구현 알고리즘

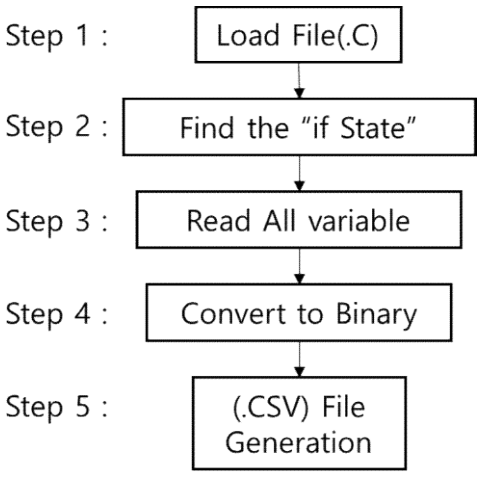

그림 4. 생성 알고리즘

구현 알고리즘은 <그림 4>와 같이 Step 1에서 테스트 소스코드가 내장된 C언어 파일을 읽어온다. 이후 Step 2에서 읽은 파일에서 if문의 조건절을 찾는다. C언어에는 if문과 else~if, else문이 세트로 조합 되어있다. else~if문도 if문에서 파생된 조건문이기 때문에 취합하면 전체 조건(변수) 확인이 가능하다. 본 연구에서는 if문과 else~if문에 사용된 조건(변수)을 "if State"로 통합하여 알고리즘에 표시하였다. 또한 if문 안에 조건은 TCAS-II에 나오는 조건식을 사용하여 알파벳 a~z까지 26개의 개별 조건으로 사용하였다.

<표 2>는 <표 1>에 사용된 TCAS-II 명세서에 중복을 제외한 개별 조건의 개수를 나타낸다.

표 2. TCAS-II 명세서 개별 조건의 개수

	Specifications	Condition Count
1	a && (!b || !c) && d || e	5
2	a && c && (d || e) && h || a && (d || e) && !h || b && (e || f)	7
3	(a && c || b && d) && e && (f || (i && (g && j || h && k)))	11
4	(a && c || b && d) && e && (f && g || !f && h)	8
5	!e && f && !g && !a && (b && c || !b && d)	7

Step 3에서는 조건문에서 사용된 조건의 개수를 중복 없이 파악한다. <표 2>에서 TCAS-II 명세서 1번의 조건식에서 사용된 조건은 a, b, c, d, e로 총 5개의 변수가 사용되었다. MCC 테스트 케이스는 파악된 조건문의 개수 N에 대해 2^N 하여 테스트 케이스 수를 생성한다. 5개의 변수로 총 32개의 테스트 케이스가 생성된다.

Step 4에서는 생성된 테스트 케이스 의 수 32를 기반으로 0부터 31(2^N-1의 값)까지 각 10진수를 2진수로 변환한다. 10진수 31을 2진수로 변환하면 $11111_{(2)}$로 총 5자리(5열)를 차지한다. 이처럼 10진수 0을 2진수로 변환 시 $00000_{(2)}$로 5자리를 맞춘다. 32개의 테스트 케이스의 경우 0부터 31행으로 총 32행, 총 5열(5자리)로 2차원 동적배열 Temp[32][5]에 2진수를 임시 저장한다. 2차원 동적배열로 설정한 이유는 TCAS-II 명세서에 사용된 개별 조건들의 개수가 다르고 Step 4처럼 각 테스트 케이스의 수를 2진수로 변환 시, 총 자릿수가 달라지기 때문이다.

2. CSV 파일생성

마지막 Step 5는 Step 4의 2차원 동적배열을 기반으로 CSV 파일을 생성한다. <그림 5>의 CSV 파일에는 생성된 조건의 개수를 위치적으로 저장한다. 조건의 개수가 10진수 10이면 2진수 $01010_{(2)}$로 CSV 파일의 11(10+1)번째 행의 위치(음영 부분)에 5자리에 맞춰서 A, B, C, D, E 열에 저장한다. 이때 2진수 1과 0은 true와 false를 의미한다. 생성된 CSV 파일은 엑셀(Excel)에서 파일정보를 확인하여 정상적으로 CSV 파일로 인식하였으며, 에러 또한 발생하지 않았다.

	A	B	C	D	E
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	0	1	1
5	0	0	1	0	0
6	0	0	1	0	1
7	0	0	1	1	0
8	0	0	1	1	1
9	0	1	0	0	0
10	0	1	0	0	1
11	0	1	0	1	0
12	0	1	0	1	1
13	0	1	1	0	0
14	0	1	1	0	1
15	0	1	1	1	0
16	0	1	1	1	1
17	1	0	0	0	0
18	1	0	0	0	1
19	1	0	0	1	0
20	1	0	0	1	1

그림 5. 생성된 MTC

Ⅴ. VectorCAST를 통한 검증

VetorCAST(2024)는 소프트웨어 동적시험 및 커버리지 분석 자동화 솔루션이며, 실제 타깃 기반 테스트 또는 코드 커버리지 분석에서 높은 신뢰성을 확보한 프로그램이다.

또한 테스트 케이스를 수동으로 생성시킬 필요 없이 CSV 파일만 있으면 테스트 케이스 데이터로 사용이 가능하다. <그림 6>은 MTC 생성기에서 생성된 CSV 파일의 테스트 케이스를 VectorCAST에서 연동시킨 화면이다. CSV 파일의 테스트 케이스 데이터 값 1은 Input Values 부분에서 true로 0은 false로 매칭되었다.

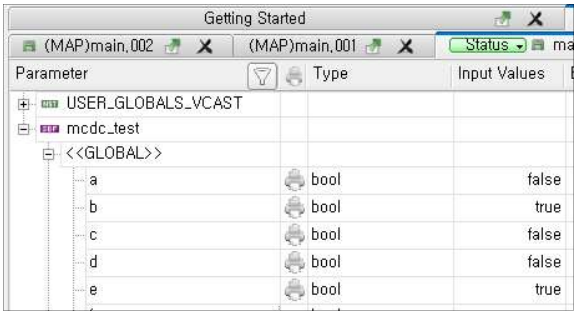

그림 6. 생성된 CSV 파일 연동 화면

표 3. TCAS-II 테스트 케이스 커버리지

	Specifications	Coverage (%)
1	a && (!b || !c) && d || e	100
2	a && c && (d || e) && h || a && (d || e) && !h || b && (e || f)	83
3	(a && c || b && d) && e && (f || (i && (g && j || h && k)))	100
4	(a && c || b && d) && e && (f && g || !f && h)	88
5	!e && f && !g && !a && (b && c || !b && d)	87

<표 3>의 Coverage(%)는 VectorCAST를 사용하여 MTC로 MC/DC 테스트를 수행한 커버리지 값이다. 이 커버리지 값은 각 명세서의 조건문이 커버될 수 있는 최댓값이다.

검증 결과 TCAS-II 명세서에 각 조건들의 최대 커버리지 값을 확인 할 수 있다. <그림 7>은 중복조건이 없는 <표 3>의 1번 조건식을 MC/DC 수행한 화면이다. 각 개별 조건이 다른 조건식에 영향을 주지 않고 전체 조건식의 결과에 독립적으로 영향을 주는 Pairs가 확인되면 최종 커버리지 값이 나온다. <그림 7>의 마지막 부분에 Pairs satisfied: 5 of 5로 5가지 개별 조건식에 대해 Pairs가 모두 만족하였으므로 커버리지 값이 100%가 나왔다.

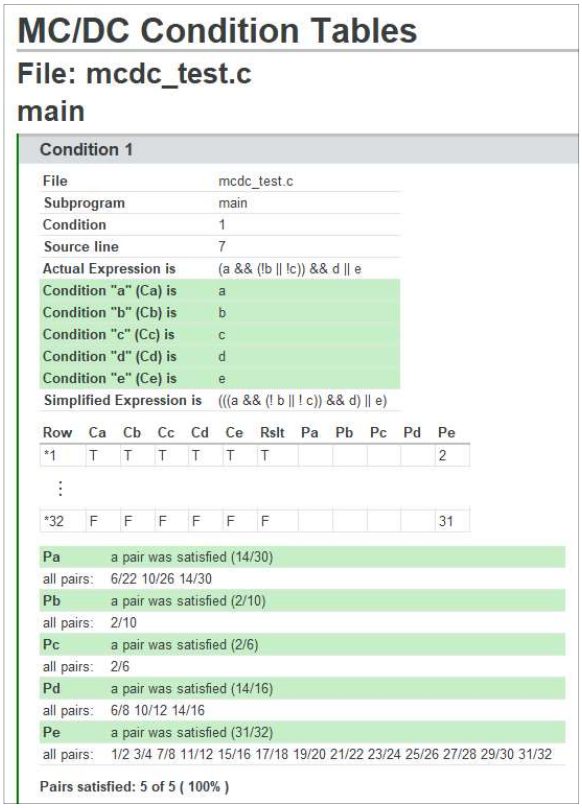

그림 7. <표 3>의 1번 명세서 조건 MC/DC 결과

<그림 8>은 중복조건이 있는 <표 3>의 2번 조건식을 MC/DC 수행한 결과이다. 중복된 개별 조건식 a와 h로 인해 Pairs를 찾지 못해 적색 음영으로 표시되었으며 최종적으로 수행된 커버리지 값은 83%가 나왔다.

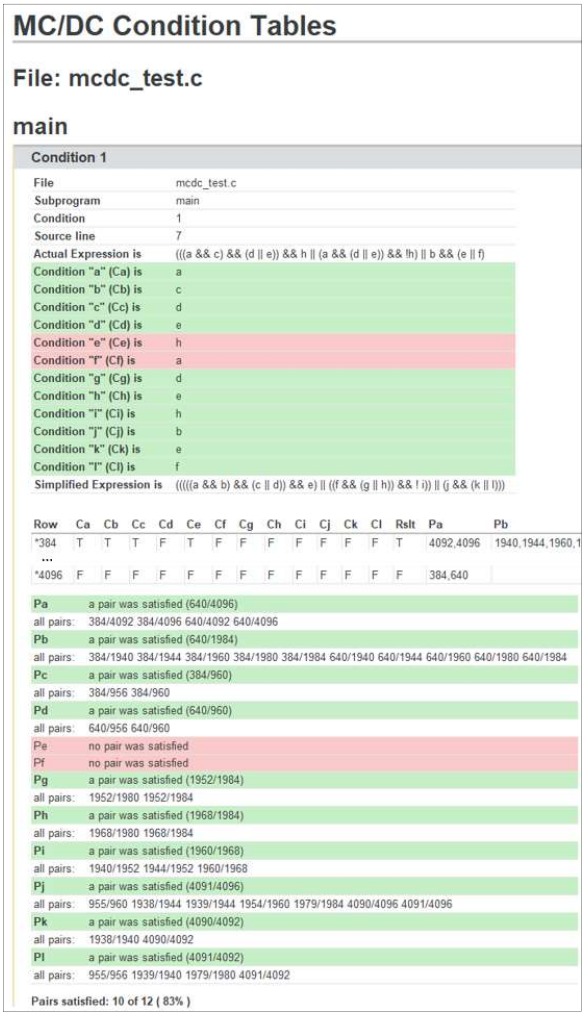

그림 8. <표 3>의 2번 명세서 조건 MC/DC 결과

Ⅵ. 결 론

본 연구에서는 MTC 생성을 위해 소프트웨어 테스트 코드가 담긴 C언어 파일을 로드하여 파싱 후 최종적으로 CSV 파일을 생성하는 프로그램을 설계하고 구현하였다. 본 연구에서 개발한 MTC 생성기의 이용은 MTC로 MC/DC를 수행하면 MC/DC의 최대 커버리지 값을 확인할 수 있다. 특히 중복이 없는 조건문은 MC/DC 커버리지 값이 100%가 됨을 확인하였다. 또한 MCC 테스트 케이스로 더 많은 테스트 케이스를 통해 결함발견 가능성이 증가한다. MCC 테스트 케이스는 테스트할 케이스 수가 많아서 대량의 테스트 케이스를 일괄적으로 수행하기 위한 파일 형식이 필요하다. 본 연구에서는 범용성 있는 CSV 파일로 생성하여 다양한 테스트 케이스 연구에 활용될 수 있게 하였다. 이를 통해 테스트 효율성 향상과 소프트웨어 품질 제고에 기여할 수 있다.

References

[1] Shekhawat, S., Iqbal, A., Srinivsan, U., Menon, P., “Automation of MC/DC Coverage Test Case Suite Deploying the Optimal Strategic Choice for Tool Development,” *ICT with Intelligent Applications Smart Innovation, Systems and Technologies,* vol. 248, pp. 433-443, Dec. 2021.

[2] Barisal, S. K., Chauhan, S. P. S., Dutta, A., Godboley, S., Sahoo, B. and Mohapatra, D. P, “BOOMPizer: Minimization and prioritization of CONCOLIC based boosted MC/DC test cases,” *Journal of King Saud University - Computer and Information Sciences,* vol. 34, no. 10, pp. 9757-9776, Nov. 2022.

[3] Barisal, S. K., Dutta, A., Godboley, S., Sahoo, B. and Mohapatra, D. P., “MC/DC guided Test Sequence Prioritization using Firefly Algorithm,” *Evolutionary Intelligence,* vol. 14, pp. 105-118, Mar. 2021.

[4] Godboley, S., Jaffar, J., Maghareh, R., Dutta, A, “Toward optimal mc/dc test case generation,” *ISSTA 2021,* pp. 505-516, Online, Denmark, Jul. 2021.

[5] Čegiň, J., Rástočný, K, “Test Data Generation for MC/DC Criterion using Reinforcement Learning,” *2020 ICSTW,* pp. 354-357, Porto, Portugal, Oct. 2020.

[6] Jaffar, J., Godboley, S., Maghareh, R., “Optimal MC/DC test case generation,” *ICSE' 19,* pp. 288-289, Montreal, Canada, May 2019.

[7] Zhipeng, Q., Lisong, W., Jiexiang, K., Zhongjie, G., Wang, Hui., Yin, W., Xiangyu, S., “A Method of Test Case Generation Based on VRM Model,” *2021 IEEE 6th ICCCS,* pp. 1099-1107, Chengdu, China, Jun. 2021.

[8] Y. Wang et al., “Uncovering Bugs in Code Coverage Profilers via Control Flow Constraint Solving,” *IEEE Transactions on Software Engineering,* vol. 49, no. 11, pp. 4964-4987, Nov. 2023.

[9] Z. Aghababaeyan, M. Abdellatif, L. Briand, R. S and M. Bagherzadeh, “Black-Box Testing of Deep Neural Networks through Test Case Diversity,” *IEEE Transactions on Software Engineering,* vol. 49, no. 5, pp. 3182-3204, May 2023.

[10] P. Singh and L. K. Singh, “Reliability and Safety Engineering for Safety Critical Systems: An Interview Study With Industry

Practitioners," *IEEE Transactions on Reliability,* vol. 70, no. 2, pp. 643-653, Jun. 2021.
[11] L. E. G. Martins and T. Gorschek, "Requirements Engineering for Safety-Critical Systems: An Interview Study with Industry Practitioners," *IEEE Transactions on Software Engineering,* vol. 46, no. 4, pp. 346-361, Apr. 2020.
[12] J. Shi, Y. Chen, Q. Li, Y. Huang, Y. Yang and M. Zhao, "Automated Test Cases Generator for IEC 61131-3 Structured Text Based Dynamic Symbolic Execution," *IEEE Transactions on Computers,* vol. 73, no. 4, pp. 1048-1059, Apr. 2024.
[13] L. Guglielmo, L. Mariani and G. Denaro, "Measuring Software Testability via Automatically Generated Test Cases," *IEEE Access,* vol. 12, pp. 63904-63916, May 2024.
[14] A. Perera, A. Aleti, B. Turhan and M. Böhme, "An Experimental Assessment of Using Theoretical Defect Predictors to Guide Search-Based Software Testing," *IEEE Transactions on Software Engineering,* vol. 49, no. 1, pp. 131-146, Jan. 2023.
[15] S. D. Semujju, H. Huang, F. Liu, Y. Xiang and Z. Hao, "Search-Based Software Test Data Generation for Path Coverage Based on a Feedback-Directed Mechanism," *Complex System Modeling and Simulation,* vol. 3, no. 1, pp. 12-31, Mar. 2023.
[16] J. Sundell, K. Lundqvist, and H. Forsberg, "Safety-critical software - quantification of test results," *2020 IEEE International Symposium on Software Reliability Engineering Workshops,* pp. 303-310, Coimbra, Portugal, Oct. 2020.
[17] E. Weyuker, T. Goradia, and A. Singh, "Automatically generating test data from a Boolean specification," *IEEE Transactions on Software Engineering,* vol. 20, no. 5, pp. 353-363, May 1994.
[18] L. Yang, J. Yan, and J. Zhang, "Generating minimal test set satisfying MC/DC criterion via SAT based approach," *Proceedings of the 33rd Annual ACM Symposium on Applied Computing,* pp. 1899-1906, Pau, France, Apr. 2018.
[19] Kitamura, T., Maissonneuve, Q., Choi, EH., Artho, C., Gargantini, A, "Optimal Test Suite Generation for Modified Condition Decision Coverage Using SAT Solving," *SAFECOMP 2018,* pp. 123-138, Vasteras, Sweden, Aug. 2018.

저 자 소 개

이로빈(정회원)

2014년 경상국립대학교 컴퓨터공학과 학사 졸업.
2016년 경상국립대학교 정보과학과 석사 졸업.
현재 경상국립대학교 컴퓨터공학과 박사과정.
<주관심분야 : 임베디드시스템, 소프트웨어 테스트>

남영호(정회원)

1994년 중앙대학교 컴퓨터공학과 박사 졸업.
현재 경상국립대학교 컴퓨터공학과교수 재직.

<주관심분야 : 임베디드시스템, 시스템 프로그래밍>