# MC/DC 100% Test case를 활용한 Back-to-Back Testing

고동률, 유영민, 박인권, 한일영
*슈어소프트테크 SE사업본부
e-mail : drko89@suresofttech.com, ymyou@suresofttech.com, inkpark@suresofttech.com, ilyounghan@suresofttech.com

# Back-to-Back Testing based on MC/DC 100% Test case

Dong-Ryul Ko*, Young-Min You, In-Kuen Park, Il-Young Han
*SW Engineering Business Division, Suresofttech

**요 약**

차량 내 전장부품이 증가하고, 차량 OEM(Original Equipment Manufacturing)이 다양한 차종을 생산 판매함에 따라 다양한 SW(software) 형상이 개발되고 있다. 따라서, 기존에 개발된 SW 형상과 변경된 SW 형상 간에 기능 일치성 검증에 대한 필요성이 증가하고 있다. 두 가지 SW 형상 간에 기능 일치성 확인을 위한 테스팅 방법으로 Back-to-Back Testing 이 있는데, 이는 각 SW 형상에 동일한 입력값을 주입하고 동일한 출력값이 산출되는 지 확인하는 테스팅 방법이다. Back-to-Back Testing 수행 시 Test case 설계가 필요한데, Test case 의 분량과 테스팅 종료기준에 대해서 아직 확립이 되어 있지 않다. 이제 본 논문에서는 MC/DC(Modified Condition /Decision Coverage) 개념을 이용하여 Test case 분량과 테스팅 종료 기준에 대해서 제시하고, 이를 적용한 사례를 설명한다. 본 논문에서 제시한 Test case 설계 기준을 적용하면, 제한적인 테스팅 일정과 인력을 만족하고, 기능 일치를 확인할 수 있는 충분한 테스팅이 가능할 것으로 판단한다.

## 1. 서론

최근 전제 자동차 제조원가에서 차지하는 전장부품 비율이 급속히 확대되고 있다. 2010 년 35%에서 2016 년에 40%를 넘어섰고, 친환경차, 자율주행 자동차 등이 활발해질 것으로 예상되는 2030 년에는 50%이상을 기록할 것으로 전망된다.[1]

한편, 다양한 소비자 요구와 친환경 자동차의 확산에 따라서 차량 OEM 이 개발하고 판매하는 차종이 늘어나고 있다.[2] 이에 차량 OEM 은 다차종 대응을 위해서 특정 전장부품에 탑재되는 SW 에 대해서 지속해서 변경과 검증을 거쳐야 한다.

이 과정에서 SW 에 새로운 기능을 추가하거나 기존 기능을 개선하는 Refactoring 을 하게 되는데, SW 기능과 신뢰성 면에서 이전 SW 와 기능이 일치하는 지 검증되어야 한다. 기존 SW 와 변경된 SW 에 동일한 입력값을 주입하여 동일한 값이 출력되는 지 검증하는 테스팅을 Back-to-Back Testing 이라고 하는데, 이 Back-to-Back Testing 은 변경된 SW 의 기능과 신뢰성을 확인하는 효과적인 검증 방법이다.[3]

Back-to-Back Testing 진행 시 필요한 입력값을 Test case 라고 하는데, 이 Test case 의 양은 이론적으로 무한대가 가능하다. 예를 들어, 실수 Type 변수의 입력값은 무한하기 때문이다. 하지만, 현실에서는 제한적인 테스팅 일정과 인력으로 인해 Test case 를 무한정하게 설계 할 수 없다. 반면, 너무 적은 Test case 의 경우 기존 SW 와 변경된 SW 의 기능이 동일함을 확인하기에 부족하다. 따라서, Back-to-Back Testing 의 Test case 분량에 대한 적절한 기준이 필요하다.

이러한 기준을 제시하기 위해서 본 논문에서는 자동차 기능안전 표준인 ISO 26262 에서 제시하는 Code coverage 개념을 적용하였다.

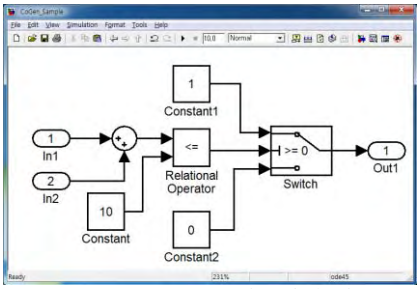
Code coverage 는 전체 Code 중에서 테스트가 수행된 Code 를 측정하여 백분율로 나타내는 개념이다. 이 개념을 통해 Test case 의 분량을 정하면, 변경된 SW 기능과 신뢰성을 확보할 정도로 충분한 Test case 라고 할 수 있으며, Code coverage 100% 달성이라는 명확한 종료기준을 세우고 테스트할 수 있어 일정 상의 예측이 가능하다.[4]

본 논문은 Code coverage 개념을 이용한 Test case 를 구축하고 이를 변경된 SW 에 적용한 사례를 제시한다. 그 전에 자동차 제어기 SW 개발환경인 MBD (Model Based Development)와 Code coverage 그리고 Back-to-Back Testing 에 대한 기본 개념을 2 절에서 설명한다.

## 2. 배경

### 2.1. MBD(Model Based Development)

기존 SW 개발 프로세스는 SW 요구사항을 정의하고 설계를 한 후, 이를 문서화하였다. 그리고 해당 문서를 기반으로 SW Code 를 개발하고 테스팅하는 과정을 거쳤으나, MBD 는 설계과정부터 Simulink(그림 1) 와 같은 Modeling tool 을 이용하여 도식화하여 설계하고, Code 자동 생성 기능을 이용하여 SW Code 를 산출하는 개발 방식이다. 도식화하여 이해하기 쉽고, 자동으로 SW Code 를 생성하기 때문에 SW 개발 생산성 관점에서 유리하다는 장점이 있다.[5]




(그림 1) Matlab Simulink 을 사용한 Model 예시

### 2.2. Code coverage

Code coverage 는 SW Code 가 얼마나 테스트되었는지를 백분율(%)로 나타내는 지표이다. ISTQB 에서는 Coverage 를 수식 1 과 같이 정의하고 있다.

$$\text{Coverage} = \frac{\textit{Number of Coverage items exercised}}{\textit{Total number of coverage items}} \times 100(\%)$$

(수식 1) ISTQB Coverage 정의[6]

Coverage 는 Code 의 실행 기준에 따라서 Statement Coverage, Branch Coverage, MC/DC 등이 가장 많이 사용된다.

1) Statement Coverage: 해당 구문이 최소 1 번 이상 실행되었는지 나타내는 비율
2) Branch Coverage: 해당 분기문에서 True/ False 조건이 최소 1 번 이상 실행되었는지 나타내는 비율
3) MC/DC: 독립적인 변화가 결과에 영향을 미치는 모든 조건에 대해 최소 1 번 이상 실행되었는지 나타내는 비율
(표 1 과 같은 경우에 4 번을 제외한 모든 경우의 조건을 만족해야 MC/DC 100%를 달성할 수 있다. )[7]

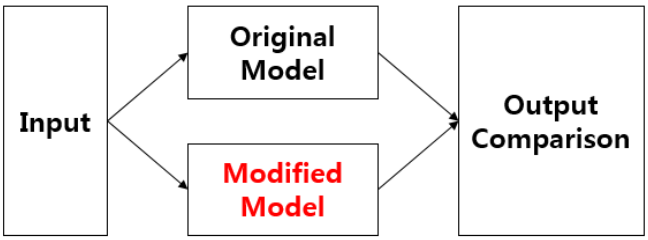
<표 1> (A && B) 조건의 경우의 수

| Case | 조건 A | 조건 B | (A And B) 결과 |
|---|---|---|---|
| 1 | True | True | True |
| 2 | True | False | False |
| 3 | False | True | False |
| 4 | False | False | False |

### 2.3. Back-to-Back Testing

ISO 26262 part4 에서 Back-to-Back Testing 은 동일한 입력에 대하여 테스트 객체와 시뮬레이션 객체의 출력을 비교하여 동작과 구현의 차이를 감지하는 것이라고 설명한다.[8]

즉, Back-to-Back Testing 은 그림 2 와 같이 2 개 이상의 객체를 동일한 입력값으로 실행하고 출력값의 일치성을 비교하는 테스팅이다. 이를 통해 변경 전 SW 와 변경 후 SW 의 기능이 일치하는지 확인한다.




(그림 2) Back-to-Back Testing 도식

## 3. MC/DC 100% Test case 를 활용한 Back-to-Back Testing 사례

이번 절에서는 Back-to-Back Testing 을 수행한 사례에 대해 기술한다. 표 2 에서 개략적인 내용을 확인할 수 있다.

<표 2> Back-to-Back Testing 개요

| | | |
|---|---|---|
| 3.1. 환경구성 | 개발 환경 | Matlab Simulink |
| | 자동화 도구 | Controller Tester |
| | 테스트 대상 | Refactoring Model |
| 3.2. Test case 설계 | MC/DC 100% 기준으로 설계 | |
| 3.3. 출력값 비교 및 Test case 재설계 | 출력값을 비교하여 일치성 확인이 가능한 Test case 설계 | |
| 3.4. 오류 분석 및 보고서 작성 | 두 SW 간 기능 확인 및 오류 분석 | |

### 3.1. 환경구성

기존 Simulink model(Model A)을 Refactoring 하여 변경된 Simulink model(Model A' )이 구축되었다. 해당 Refactoring 은 기존 Model 의 기능 변화 없이 구조와 성능 측면에서의 개선을 목적으로 하였다.

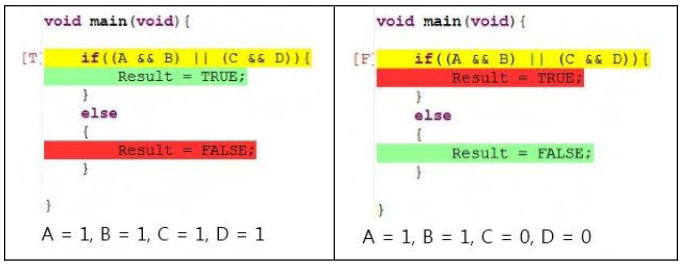
이 2 종의 Model 에 대해서 Code 자동 생성기로 C code 2 종(Code A, Code A' )을 생성하여 테스팅을 착수할 수 있는 환경을 갖추었다.

### 3.2 Test case 설계

먼저 Code A 를 분석하여 Test case 를 설계하였다. 이 Test case 는 MC/DC 100%를 만족하는 Test case 이며, 이를 Code A' 에 적용하게 된다.

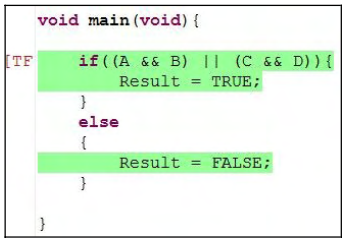
Test case 설계와 테스트 수행, 테스트 과정에서의 Code coverage 측정은 테스팅 자동화 도구인 슈어소프트 사의 Controller Tester 를 사용하였다.

Test case 설계 과정을 상세히 예를 들면, 그림 3 의 좌측과 같이 작성한 Test case 의 결과가 조건문의 True 만 만족한다면, 그림 3 의 우측과 같이 False 를 만족할 수 있는 Test case 를 작성한다.




(그림 3) True 조건과 False 조건

True 와 False 조건을 모두 만족하여 그림 4 와 같은 결과를 확인하면, Branch Coverage 를 달성하는 Test case 를 작성한 것으로 판단한다.




(그림 4) Branch Coverage 만족 예시

그림 4 와 같은 예시를 대상으로 MC/DC 100%를 달성하려면, 그림 5 의 경우의 수를 모두 만족시켜야 한다.

| ((A&&B)\|\|(C&&D)) | A | B | C | D |
|---|---|---|---|---|
| True | True | True | Don't care | Don't care |
| True | True | False | True | True |
| False | True | False | True | False |
| False | True | False | False | Don't care |
| True | False | Don't care | True | True |
| False | False | Don't care | True | False |
| False | False | Don't care | False | Don't care |

(그림 5) MC/DC 진리표

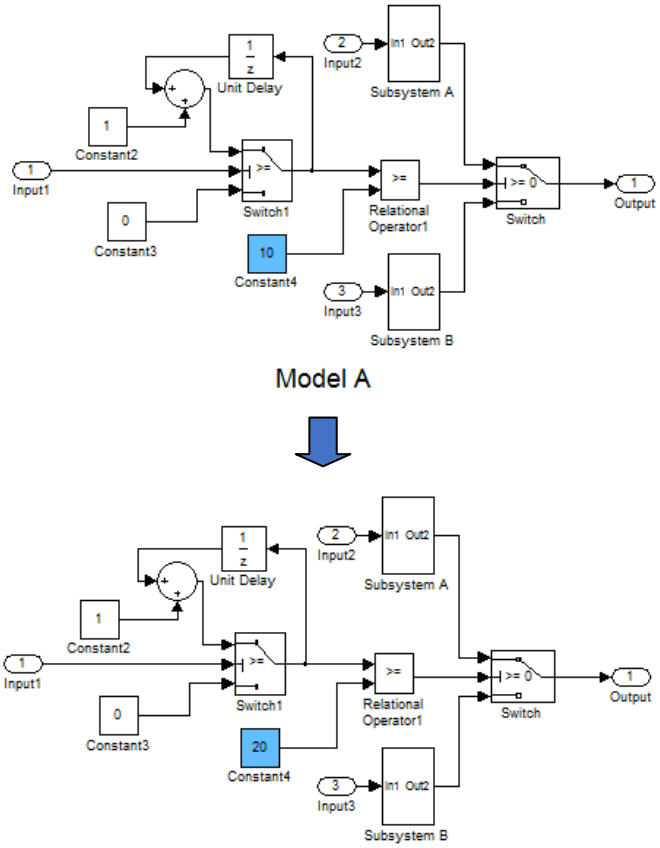
Code A 를 대상으로 위와 같은 작업을 반복하여, MC/DC 100% 달성을 기준으로 Test case 를 설계하였다. 여러 개의 Test case 를 Set 로 묶어서 모든 Coverage 를 확인하는 하나의 Test suite 을 구축하였다.

### 3.3 출력값 비교 및 Test case 재설계

확보한 Test suite 을 Code A' 에 적용하여 테스트하고 출력값을 Code A 의 출력값과 비교하였다.

값이 다르게 출력된다면, 원인을 분석하여 Test suite 을 다시 설계한다. 변경된 Logic 이 다른 Logic 에도 영향을 미칠 수 있기 때문에, 변경된 Logic 을 우회하는 Test suite 을 설계해서 다시 테스트해야 한다.

예를 들어, 총 100 개의 Test case 로 Test suite 을 구성하여 테스트했다고 가정했을 때, 50 번째 Case 의 출력값이 기댓값과 다르다면, 51 ~ 100 번째 Case 는 신뢰할 수 없다. 이러한 상황을 그림 6 을 통해서 자세히 설명하겠다.




(그림 6) Logic 변경 예시

위 그림 6 의 Model A 는 Boolean Type 의 Input1 이 10 번 연속으로 1 이 입력되면, Subsystem A 를 수행하고 10 번 보다 적으면 Subsystem B 를 수행하는 Logic 이다.

이 Logic 에서 Model A' 과 같이 비교 조건의 상수값이 20 으로 잘못 변경되었다면, 10 번 연속으로 1 을 입력하여 50 번째 Test case 에서 Subsystem A 가 수행되도록 설계했지만, Model A' 에서는 20 번 연속으로 입력해야 하기 때문에 Subsystem B 가 수행된다.

그리고 Subsystem A 를 테스트하기 위해 설계했던 50 번째 이후에 Test case 에서도 문제가 발생한다.

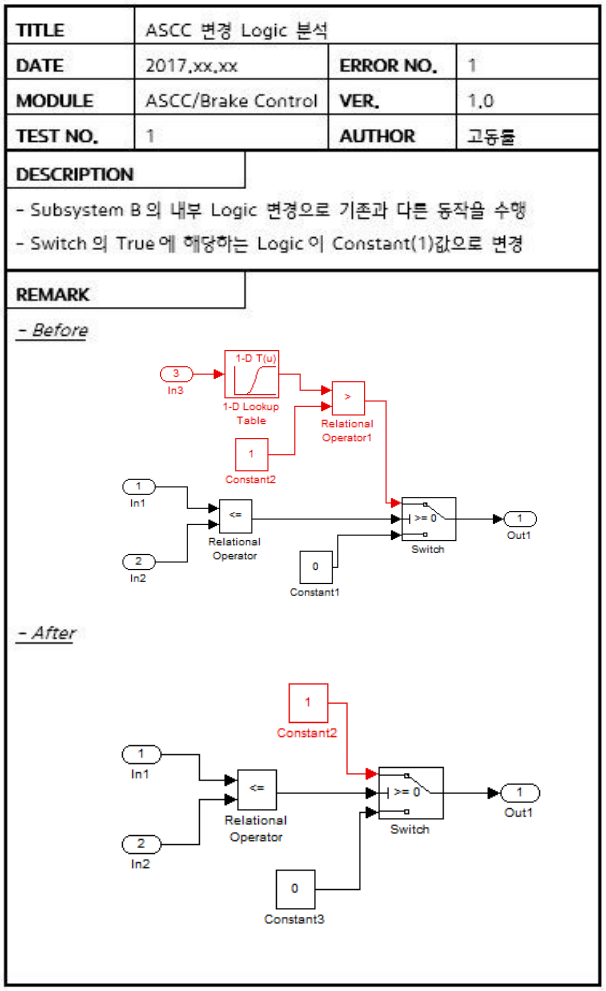
이러한 이유 때문에 100 개의 Test case 로 구성된 Test suite 에서 50 번째 Case 가 틀렸다면, 51~100 번째 Case 는 신뢰할 수 없다.

따라서 잘못 변경된 조건을 우회하는 Test suite 을 설계하고 다시 테스트해야 한다.

### 3.4 오류 분석 및 보고서 작성

3.2 와 3.3 의 단계를 반복하여 변경 전후의 일치성을 확인할 수 있는 Test suite 을 확보했다면, 두 Model 에서 생성된 Code 의 출력값을 비교하여 기능상 일치하는 않는 부분을 검출한다. 만약 출력값이 다르다면, 해당 변수와 Code, Model 을 분석하여 기능상 다른 부분을 확인하고 Model 을 수정한다.

Back-to-Back Testing 수행 후 기능 상 틀린 점이 발생하는 경우 그림 7 처럼 보고할 수 있다.




(그림 7) Back-to-Back Testing 보고서 예시

Model 수정 후에 다시 Code 를 자동 생성한 후, 해당 Test suite 으로 Back-to-Back Testing 을 다시 수행한다. 이 작업을 반복 수행해서 Model A 의 MC/DC 100% Test suite 으로 Model A' 를 수행하였을 때, 두 Model 의 출력값이 모두 일치하면 테스트를 종료한다.

## 4. 결론

앞서 얘기한 것처럼 차량용 제어기 SW 는 양적으로 질적으로 확대되어 가고 있다. 다양하고 급변하는 소비자의 요구와 시장환경에 맞춰 SW 의 변경 빈도도 급증할 것이다.

급증하는 SW 의 변경 빈도에 따라서 SW 의 신뢰성과 기능을 확인할 수 있는 Back-to-Back Testing 의 중요성이 커진다. 또한, 급증하는 빈도에 맞춰 빠르고 정확한 테스팅이 진행되어야 하며, Test case 분량에 대한 기준도 필요하다. 이에 ISO 26262 기준을 차용한 Coverage 100%의 Test case 기준은 제한된 테스팅 일정과 인력을 만족하면서 충분한 기능 검증이 가능하다고 할 수 있다.

ISO 26262 에서는 ASIL(Automotive Safety Integrity Level)에 따라서 권고하는 Coverage 의 종류가 다르다. 그 중 MC/DC 는 독립적인 세부 조건까지 모두 따지기 때문에, 안전이 매우 중요한 최고 등급 ASIL D 에서 권고한다. 따라서 MC/DC 는 변경 된 SW 간에 일치성을 비교하는 Test case 설계방식으로 적합하다.

향후에는 팀 단위로 SW 를 나눠서 개발을 하고 초기에 자주 통합하는 CI(Continuous Integration) 개발방식에 MC/DC 100% Test case 기반 Back-to-Back Testing 적용을 연구하고자 한다. 각자 개발한 Code 를 하나로 통합할 때, 개발한 Code 이외 부분에서 문제가 발생하는지 확인하는 용도로 사용할 수 있을 것으로 예상한다. 또한 젠킨스와 같은 CI 도구를 사용하면 통합할 때, Test case 를 자동으로 실행하여 업무 생산성을 높일 수 있을 것으로 기대한다.[9][10]

## 참고문헌

[1] 한국전기자동차협회, "자동차와 ICT 의 '만남과 대결'… 지각변동 진원지 전장기술", 04 January (2017), (http://www.keva.or.kr)
[2] 환경부, "친환경 자동차", pp.42-47,
[3] Wendy W. Peng, Dolores R. Wallace, "Software Error Analysis", pp. A-1
[4] Martin Fowler, "Test Coverage", 17 April (2012), (https://martinfowler.com)
[5] B. Schätz, A. Pretschner, F. Huber, and J.Philipps, "Model-Based Development of Embedded Systems", pp. 2
[6] ISTQB EXAM CERTIFICATION, "What is test coverage in software testing? It's advantages and disadvantages" (http://istqbexamcertification.com)
[7] 이성배, 심정민, 한일영, "국산 시험 자동화 도구를 활용한 무기체계 SW 신뢰성 시험사례" 한국 2016.10 정보과학회지 pp. 62
[8] ISO 26262 part-4, Table 5
[9] Jenkins (https://jenkins.io)
[10] Martin Fowler, "Continuous Integration", 01 May (2006), (https://martinfowler.com)
[11] ISO 26262 part-6: Road vehicles-Functional safety-Part 6: Product development at the software level