도형 요구사항의 MC/DC 테스트 케이스 생성 기법

원종섭*, 최경희*, 정기현**
*아주대학교 정보통신전문대학원
**아주대학교 전자공학부
e-mail : jongseob@gmail.com

Generation Technique of MC/DC Test Cases for Requirement Diagrams

Won Jong-Seob*, Choi Kyung-Hee*, Jung Ki-Hyun**
*Graduate School of Information and Communication, Ajou University
**Division of Electronics Engineering, Ajou University

요 약

본 논문은 도형 요구사항(Requirement Diagram)을 기반으로 MC/DC 를 만족하는 테스트 케이스를 생성하는 방법을 기술한다. MC/DC 는 테스트 케이스의 여러 커버리지 중 테스트 케이스의 질 측면과 효율성 측면에서 우수하다. 하지만 MC/DC 는 구조적 커버리지(Structural Coverage) 로써 이를 바로 도형 요구사항에 대해 적용시킬 수 없다. 그러므로 MC/DC 를 기반으로 도형 요구사항에 적용하는 방법을 제시한다.

1. 서 론

테스트 케이스 생성은 소프트웨어의 질을 결정하는 테스트 작업에서 매우 중요한 부분을 차지하는 작업이다. 테스트의 개수가 적으면 테스트의 질이 낮아지고, 개수가 많아지면 테스트에 많은 비용이 소모된다. 따라서, 소프트웨어의 질을 높이면서 최적의 테스트에 적합한 커버리지의 테스트 케이스를 생성하는 것이 매우 중요하다.

적절한 개수의 테스트 케이스를 생성하는 기법으로 MC/DC 기법이 있다. MC/DC[1]는 구조적 커버리지(Structural Coverage)로 시스템 내부의 조건(Condition)과 결정(Decision)을 기반으로 테스트 케이스를 생성하는 방법이다. 매우 효과적인 테스트 케이스를 생성하기 때문에 많은 임베디드 소프트웨어의 구조적 테스트 케이스 생성에 사용되고 있다.

본 논문에서는 도형으로 기술된 요구사항의 MC/DC 를 만족하는 테스트 케이스를 생성하는 방법을 기술한다. 도형 요구사항은 요구사항을 명확하며 기술하기 위하여 요구사항을 그림 등 도형으로 기술하는 방법이다. 일반적으로 자연어로 기술된 요구사항은 의미가 분명하지 않아 조건이나 결정에 모호성이 존재하여 MC/DC 기술을 적용하기 어렵다. 그러나, 의미가 분명한 도형 요구사항은 마치 소스 프로그램처럼 의미가 분명하여 MC/DC 기법을 적용할 수 있다. MC/DC 를 기반으로 도형 요구사항에 적용하는 방법에 대해서 설명한다

2 장에서는 테스트에 관한 관련연구 및 MC/DC 를 설명한다. 3 장에서 도형 요구사항을 설명하고, 4 장에서 MC/DC 를 만족하는 테스트 케이스 생성 방법을 설명한다.

2. 관련연구

2.1 테스트 케이스 생성 기법

테스트 케이스를 생성하는 기법에는 구조적 테스트 케이스 생성 방법과 Black-box 테스트 케이스 생성 기법으로 나뉘어 진다.

이쌍 테스팅(Pairwise testing)은 Black-box 테스트 케이스 생성기법의 하나로 입력의 조합을 이용하여 테스트 케이스를 생성하는 방법이다[2]. 이 방법은 매우 적은 테스트 케이스를 이용하여 효과적으로 테스팅을 가능케 한다. 그러나 시스템의 입력들만을 고려하여 테스트 케이스를 생성하기 때문에 시스템 내부 관련성을 고려해야 하는 구조적 테스트에는 적합하지 않다. 모듈 의존성을 기반으로 이쌍 테스팅의 테스트 케이스 생성 방법은 그러한 문제를 해결하기 위한 방법으로 내부 함수 관련성을 기반으로 테스트케이스를 생성하는 방법을 제안되었다[3]. 하지만 이 방법 또한 함수간 관련성만을 고려하여 테스트 케이스를 생성하므로 함수 내부 관련성은 고려하지 않는다.

따라서 이 문제를 해결하기 위해서는 함수 내부의 관련성을 고려하는 MC/DC 테스트 케이스를 생성함으로써 해결될 수 있다. MC/DC 는 구조적 테스트 케이스 생성 방법에 속한다.

2.2 Modified Condition/Decision Coverage

MC/DC 는 구조적 커버리지 중 하나로 시스템 내부의 조건과 결정을 기반으로 테스트 케이스를 생성한다. MC/DC 는 일반적으로 구조적 커버리지 중 비용과 효율성 측면에서 가장 유용하다고 알려져 있다[1].

MC/DC 의 정의는 다음과 같다.

1. 모든 진입지점과 종료지점이 한번 이상 방문되어야 한다.
2. 모든 결정의 가능한 값들이 한번 이상 출현해야 한다.
3. 조건의 가능한 결과들을 한번 이상 출현해야 한다.
4. 각 결정은 조건에 독립적으로 영향을 끼쳐야 한다.

예를 들어, 참, 거짓을 가질 수 있는 불린 매개변수 X, Y 에 대해서 식 X & Y 에 대한 테스트 케이스를 생성한다고 가정하자. MC/DC 를 만족하는 테스트 케이스는 다음과 같다.

<표 1> MC/DC 를 만족하는 테스트 케이스들

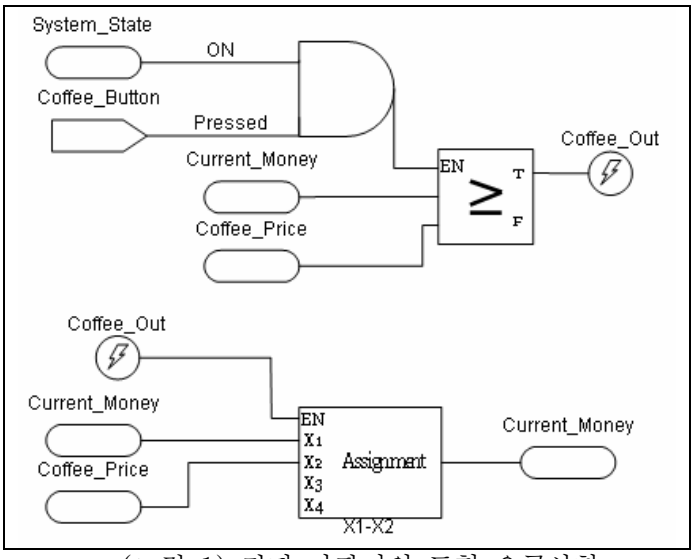
순번	X	Y	출력
1	참	참	참
2	참	거짓	거짓
3	거짓	참	거짓

위의 MC/DC 정의에 따라서 <표 1>의 테스트 케이스가 MC/DC 를 만족하는지 확인해보자. 1 번 조건은 테스트 케이스를 생성하는 과정에서 자동으로 만족된다. 2 번 조건은 X 의 모든 가능한 값 참, 거짓을 가지고 Y 역시 모든 가능한 값 참, 거짓을 가지므로 만족한다. 3 번 조건은 출력이 참 그리고 거짓을 가지므로 만족한다. 마지막으로 4 번 조건은 1, 2 번 테스트 케이스에서 Y 를 참으로 고정시키고 X 를 참에서 거짓으로 변경시켰을 때 출력이 참에서 거짓으로 변경되었다. 그리고 1, 3 번 테스트 케이스에서 X 를 참으로 고정시키고 Y 는 참에서 거짓으로 변경시켰을 때, 출력이 참에서 거짓으로 변경됨을 알 수 있다. 즉, 각 입력이 결과에 독립적으로 영향을 끼쳤다. 따라서, 4 번째 조건을 만족된다. 이와 같은 방법으로 다른 블록에 대해서도 MC/DC 의 조건을 만족하는 테스트 케이스 생성 방법을 적용했다.

3. 도형 요구사항(Requirement Diagram)

도형 요구사항은 시스템 모델이 될 수 있을 정도로 자세하게 시스템을 기술할 수 있고 실행도 가능하다. 다음 (그림 1)은 도형 요구사항으로 작성된 커피 자판기 모델의 일부분이다.

(그림 1)의 커피 자판기는 전체 모델 중 일부분으로 커피 버튼을 눌렀을 경우 현재 삽입된 가격을 비교하여 커피 가격보다 많거나 같은 경우 커피가 나오게 하고 커피 한잔의 가격을 차감하여 현재가격을 다시 계산하는 부분이다.



(그림 1) 커피 자판기의 도형 요구사항

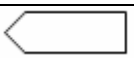
좀더 자세히 설명하면, 상단의 도형 요구사항을 보면 System_State 가 ON 이고, Coffee_Button 이 Pressed 상태일 때, Current_Money 와 Coffee_Price 의 값을 비교하여 Current_Money 의 값이 더 크다면, Coffee_Out 이벤트를 발생시킨다. 하단의 도형 요구사항을 보면 Coffee_Out 이벤트가 발생하면, Current_Money – Coffee_Price 를 계산하여 Current_Money 에 저장한다.

도형 요구사항은 (그림 1)과 같이 시스템을 직관적으로 모델링 할 수 있다.

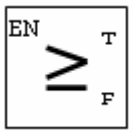
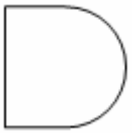
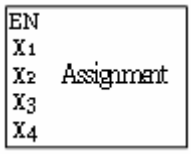
도형 요구사항은 엔티티(Entity) 블록과 조작(Operational) 블록으로 구분된다[4]. 엔티티 블록은 처음 부분 또는 말단에 오는 블록으로써 (그림 1)의 System_State, Coffee_Button, Current_Money, Coffee_Price, Coffee_Out 을 예로 들 수 있다<표 1>. 조작 블록은 중간에 나오는 블록으로 (그림 1)의 Flow_AND, Assignment, Greater Than Equal 와 같은 블록을 말한다<표 2>.

블록들은 포트가 존재하며 블록들은 포트를 통하여 다른 블록과 연결된다. 포트와 포트는 링크로 연결된다. 각 포트 또는 링크는 활성화 또는 비활성화 될 수 있다. 활성화는 해당 포트로 값이 전달된 다는 의미이고, 비활성화는 값이 전달되지 않는다는 것이다.

<표 2> 엔티티 블록 설명

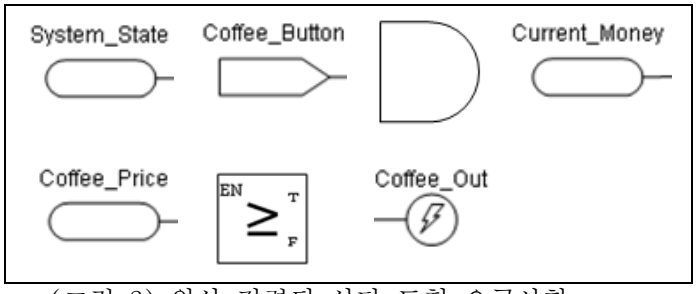
그림	이름	설명
	Memory	Variable 을 의미한다. 정수, 실수, 문자열 형의 값을 저장할 수 있다.
	Input Device	버튼 등 외부 입력을 받는 블록이다.
	Output device	외부로 값을 전달할 때 사용하는 블록이다.
	Internal Event	내부적으로 사용되는 이벤트 블록이다.

<표 3> 조작 블록 설명

그림	이름	설명
	Greater than Equal	EN 포트가 활성화되었을 때, 첫 번째 포트로 입력된 값이 두 번째 포트에서 들어온 값보다 크거나 같은 경우 T 포트가 활성화되고 F 포트는 비활성화된다. 그렇지 않으면 F 포트가 활성화되고 T 포트는 비활성화된다.
	Flow_ AND	입력 포트들이 모두 활성화된 경우 출력 포트가 활성화된다.
	Assign ment	모든 입력 포트가 활성화되었을 때, X1~X4 로부터 입력된 값이 Assignment 하단에 있는 식을 만족한 경우 출력 포트가 활성화된다.

4. MC/DC 를 만족하는 테스트 케이스 생성 방법

(그림 1) 커피 자판기의 상단 도형 요구사항에서 테스트 케이스는 다음과 같은 순서로 생성된다. 도형 요구사항은 블록들에 대해서 위상 정렬 후, 차례대로 테스트 케이스를 생성한다. 상단의 도형 요구사항에 대해서 정렬된 블록들의 모습은 다음과 같다(그림 2).



(그림 2) 위상 정렬된 상단 도형 요구사항

모든 블록은 각각 MC/DC 를 만족하는 테스트 케이스 생성 알고리즘을 정의하고 있다. 엔티티 블록은 해당 엔티티에 대해서 미리 정의된 대표 값들을 이용하여 테스트 케이스를 만든다. 이 테스트 케이스는 다음 블록으로 전달되어 다음 블록에서 테스트 케이스를 생성하는데 사용된다. System_State 는 ON, OFF 두 개의 대표 값들을 갖는다. 이러한 대표 값들은 사용자에 의해서 입력된다. ON 은 해당 링크를 활성화시키고 OFF 는 비활성화시킨다. 이 정보가 다음 블록(Flow_AND)에 전달된다. Coffee_Button 은 Pressed 와 Unpressed 두 개의 대표 값을 이용하여 다음 블록(Flow_AND)에 테스트 케이스를 전달한다.

엔티티 블록에서 생성된 테스트 케이스는 조작 블록에 전달된다. 조작 블록은 미리 정의된 MC/DC 를 만족하기 위한 알고리즘을 이용하여 테스트 케이스를 만든다. Flow_AND, Greater Than Equal 는 MC/DC 생성 방법과 동일한 방법으로 생성되지만, 그 밖에 MC/DC 생성 방법으로 테스트 케이스를 만들 수 없는 블록들에 대해서는 새로운 방법이 고안되어 테스트 케이스를 생성한다.

System_State 와 Coffee_Button 으로부터 테스트 케이스를 받은 Flow_AND 는 다음과 같은 테스트 케이스를 만든다<표 4>.

<표 4> Flow_AND 블록에서 테스트 케이스 생성

순번	System_State	Coffee_Button	Output
1	ON	Pressed	활성화
2	OFF	Pressed	비활성화
3	ON	Unpressed	비활성화

Current_Money 은 100, 200, 500 총 세 개의 대표 값을 갖는다. Coffee_Price 는 200 한 개의 대표 값을 갖는다. 두 엔티티 블록은 해당 대표 값을 이용하여 테스트 케이스를 생성한 후 이를 다음 블록에 전달한다. Greater Than Equal 블록은 앞에서 만들어진 테스트 케이스들을 조합하여 <표 5>의 테스트 케이스를 생성된다. Coffee_Out 은 말단 엔티티 블록이므로 테스트 생성에 관여하지 않는다. <표 5>는 따라서 최종 테스트 케이스이다.

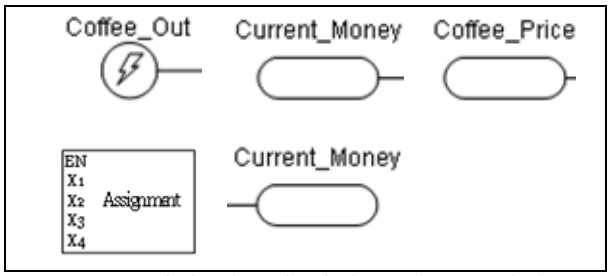
<표 5> 상단 도형 요구사항의 최종 테스트 케이스

순번	System_State	Coffee_Button	Current_Money	Coffee_Price	Coffee_Out
1	ON	Pressed	500	200	1
2	ON	Pressed	300	200	1
3	ON	Pressed	100	200	0
4	OFF	Pressed	500	200	0
5	OFF	Pressed	200	200	0
6	ON	Unpressed	500	200	0
7	ON	Unpressed	200	200	0

생성된 테스트 케이스 결과를 분석해보면 다음과 같다. 모든 블록이 방문되었으니 MC/DC 의 첫 번째 조건이 만족되었다. 두 번째 조건에 대해서 분석하면, System_State 는 가질 수 있는 모든 값 ON, OFF 를 가진다. Coffee_Button 또한 가질 수 있는 모든 값 Pressed, Unpressed 가 나타났다. Current_Money 와 Coffee_Price 또한 각 엔티티가 가질 수 있는 모든 값 500, 300, 100 모두 출현한다. 따라서, MC/DC 의 두 번째 조건을 만족한다. 출력인 Coffee_Out 을 보면 가질 수 있는 모든 값 1, 0 를 가진다. 1 은 이벤트가 발생한 것을 나타내며, 0 은 이벤트가 발생하지 않은 것을 의미한다. 따라서, 세 번째 조건을 만족한다.

마지막으로 네 번째 조건을 검토해보면, System_State 를 ON, Coffee_Button 을 Pressed, Coffee_Price 를 200 으로 고정시키고 1, 3 번 테스트 케이스를 비교하면 Current_Money 가 500 에서 100 으로 변경됨에 따라 Coffee_Out 이 1 가 0 로 변경되었다. 이와 같은 방법으로 System_State(1,4 번), Coffee_Button(1, 6 번)에 대해서 검사했을 때, Coffee_Out 이 변경되었다. Coffee_Price 는 한 가지의 값 만을 가지므로 고려하지 않는다. 따라서, MC/DC 의 네 번째 조건을 만족한다.

(그림 1)의 커피 자판기의 하단의 도형 요구사항에서 테스트 케이스는 다음과 같은 순서로 생성된다. 도형 요구사항은 블록들에 대해서 위상 정렬한 후, 차례대로 테스트 케이스를 생성한다. 하단의 도형 요구사항에 대해서 정렬된 블록들의 모습은 다음과 같다(그림 3).



(그림 3) 위상 정렬된 하단 도형 요구사항

Coffee_Out 이벤트 블록은 1 또는 0 의 대표 값을 가진다. 1 은 해당 포트를 활성화 시키고, 0 은 해당 포트를 비활성화시킨다. Current_Money 는 100, 200, 500 총 세 개의 대표 값을 갖는다. Coffee_Price 는 200 한 개의 대표 값을 갖는다. 대표 값들은 사용자에 의해서 입력된 값이다. Assignment 는 이전 블록들로부터 입력된 테스트 케이스를 받아 최종 테스트 케이스를 생성한다. 말단의 Current_Money 는 테스트 케이스 생성에 관여하지 않는다.

<표 6> 하단 도형 요구사항의 최종 테스트 케이스

순번	Coffee_Out	Current_Money	Coffee_Price	Current_Money(결과)
1	1	100	200	-100
2	1	200	200	0
3	1	500	200	300
4	0	500	200	(변화없음)
5	0	200	200	(변화없음)
6	0	100	200	(변화없음)

하단 도형 요구사항에 대해서도 생성된 테스트 케이스 결과를 분석해보면 다음과 같다. 모든 블록이 방문되었으니 MC/DC 의 첫 번째 조건이 만족되었다. 두 번째 조건에 대해서 분석하면, Coffee_Out 은 가질 수 있는 모든 값 1, 0 값을 가진다. Current_Money 또한 가질 수 있는 모든 값 100, 200, 500 모두 출현했다. Coffee_Price 또한 각 엔티티가 가질 수 있는 모든 값 200 이 출현한다. 따라서, MC/DC 의 두 번째 조건을 만족한다. 출력인 Current_Money(결과)를 보면 가질 수 있는 모든 값 1, 0, (변화없음)을 가진다. 따라서, 세 번째 조건을 만족한다. 마지막으로 네 번째 조건을 검토해보면, 1, 2 번 테스트 케이스를 비교하면 Coffee_Out 은 1, Coffee_Price 를 200 으로 고정시키고 Current_Money 를 100 에서 200 으로 변경시켰을 때 Current_Money(결과)는 -100 과 0 으로 결과가 다르다. 그리고 1,6 번 테스트 케이스에서는 Current_Money 를 100 으로 Coffee_Price 는 200 으로 고정시키고, Coffee_Out 을 1 과 0 을 입력했을 때, Current_Money(결과)는 -100 과 (변화없음)으로 다른 결과가 도출되었다. Coffee_Price 는 한 가지의 값 만을 가지므로 고려하지 않는다. 따라서, 검토해본 결과 MC/DC 의 네 번째 조건을 만족한다.

5. 결론 및 추후 연구 과제

본 논문에서는 도형 요구사항에 대해서 MC/DC 를 만족하는 테스트 케이스를 생성 기법을 제안하였다. 현재 대표 값은 사용자에 의해 입력된다. 사용자 편의성과 정확한 테스트 케이스 생성을 위하여 대표 값은 자동으로 생성되어야 한다. 마지막으로 생성된 테스트 케이스가 실행 가능한지 여부에 대한 연구가 필요하다. 예를 들어, <표 6>의 1 번 테스트 케이스는 Current_Money 가 Coffee_Price 보다 작은 경우, Coffee_Out 이벤트는 절대 발생하지 않기 때문에, 해당 테스트 케이스는 실행 가능하지 않다.

본 연구는 산업자원부 및 한국부품소재산업진흥원의 부품소재개발사업의 연구결과로 수행되었음

참고문헌

[1] Kelly Hayhurst, et al, “A Practical Tutorial on Modified Condition/Decision Coverage”, NASA/TM-2001-210876, May 2001,

[2] K.-C. Tai and Y.Lei, “A Test generation strategy for Pairwise testing,” IEEE Trans. Softw. Eng., vol.28, no.1, pp.109-111, Jan. 2002.

[3] Jangbok Kim, Kyunghee Choi, and Gihyun Jung, "Pairwise test case generation based on module dependency", IEICE Transaction Information and System, Vol.E89-D No.11 pp.2811-2813 November 01, 2006

[4] 이혜련, 최경희, 정기현, “Requirement Diagram을 자연어로 작성하기 위한 Translation Database Design”, 제 28회 한국정보처리학회 추계학술발표대회 논문집 제 14권 제 2호, 2007.11