테스트 케이스 생성을 위한 그래픽 언어 기반 요구사항 다이어그램의 상태 공간 변환 기법

신상기*, 정기현*, 최경희**
*아주대학교 전자공학부
**아주대학교 정보통신전문대학원
e-mail : gofolder@gmail.com

State Space Translation technique of Requirement Diagram based Graphical Language for Test Case

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

요 약

그래픽 언어 기반의 요구사항 작성은 임베디드 시스템의 RBT(Requirement-Based Testing)을 위해 적절한 요구사항 모델링이다. 본 논문은 그래픽 언어기반으로 작성된 요구사항을 상태 기반으로 해석하고, 테스트 케이스를 자동으로 생성하기 위한 방법에 대해 논한다. 먼저 요구사항 다이어그램의 구성을 살펴 보고, REED 라는 도구를 통해 실제 작성된 하나의 요구사항을 바탕으로 테스트 케이스 생성을 위한 모델 전체의 상태 공간을 정의하고 상태 공간과 테스트 케이스 생성에 대해 논한다.

1. 서론

임베디드 시스템의 높은 품질에 대한 기대로 테스팅의 중요성은 커지고 있다. 높은 신뢰수준을 요구하는 항공이나 의료 기기의 임베디드 시스템의 경우, V&V(validation and verification)단계는 전체 프로젝트 개발 자원의 50%~70%를 차지하기도 한다[3]. 테스트를 위한 일련의 작업들 속에서 테스트 케이스 생성은, 테스트의 질과 효율을 결정하는 중요한 작업이다. 테스팅 자동화를 위한 다양한 툴들이 시장에서 쓰이고 있지만[4,5,6] Requirement-Based Testing(RBT)[8]를 위한 만족 조건 중 테스트 가능한 형태로 요구사항이 작성될 수 있어야 한다는 조건을 기존의 툴들이 항상 만족하지는 못한다. 그래픽 언어 기반으로 작성된 요구사항은 이러한 문제점을 해결하여, 단일 요구사항에 대해 하나의 요구사항 다이어그램으로 표현이 가능하다[1]. 본 논문은 그래픽 언어를 이용한 요구사항 모델로부터 사용자가 요구하는 커버리지를 만족하는 테스트 케이스를 자동으로 생성하는 방법에 대해 기술한다.

2 장에서는 본 논문과 관련된 연구를 살펴 보고 3 장에서는 그래픽 기반의 요구사항 모델링 도구인 REED 로 표현한 요구사항 다이어그램에 대해 알아본다. 4 장에서는 테스트 케이스 생성 절차와 모델의 상태 공간을 구하기 위한 기법을 살펴본다. 마지막으로 5 장에서는 결론과 향후 과제에 대해 언급한다.

2. 관련 연구

2.1 상태 모델 기반의 테스트 케이스 생성 기법

상태 다이어그램 기반으로 기술된 요구사항 모델에 대해 모델 검증 도구를 이용하여 테스트 케이스를 생성하는 기법들이 있다. RSML(Requirements State Machine Language)[9]을 이용하여 기술한 요구사항 모델은, Model checker[10]를 통해 주어진 property 의 부정을 입증하는 counterexample 을 생성 할 수 있다. 테스트 케이스 생성을 위해 property 들은 결과로 얻게 되는 counterexample 이 테스트 케이스로 활용 될 때, 일정한 커버리지를 만족하게 구성한다. 결과적으로 Model checker 가 생성하는 counterexample 은 주어진 커버리지를 만족하는 테스트 케이스로 활용할 수 있다[3,7]. model checker 는 요구사항 모델이 명시적으로 상태 기반으로 모델링 되어 있는 경우, 모델의 상태 변화를 실행 트리로 구성할 수 있다. 실행 트리에서 property 의 부정을 만족하는 상태는, 초기 상태로부터 해당 상태에 도달 할 수 있는 경로가 원하는 테스트 케이스가 된다.

2.2 그래픽 언어 기반의 요구사항 모델의 테스트 케이스 생성 기법

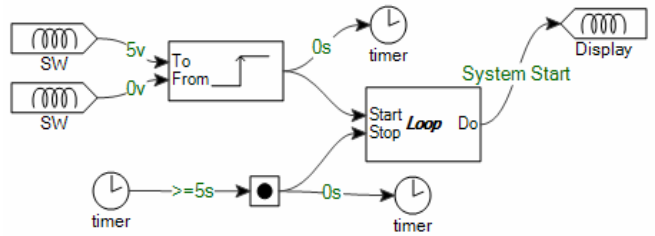
그래픽 기반의 요구사항 모델에 대한 테스트 케이스 생성 기법으로, 요구사항 다이어그램의 정적인 분석을 통해 테스트 케이스를 생성하는 연구가 진행 되었다[2]. 이 기법은 요구사항 다이어그램의 블록들을 위

상 정렬 하고, 사용자에 의해 입력된 대표 값을 이용하여 테스트 케이스를 생성하는 기법이다. 이 연구에서는 그래픽 언어로 표현된 요구사항 다이어그램이 커버리지 중 가장 질 측면과 효율성 측면에서 우수한 MC/DC 를 만족하는 테스트 케이스 생성이 가능함을 보였다. 하지만, 정적인 분석을 통한 테스트 케이스 생성 기법은 복잡한 구조의 요구사항 모델에 대해 실제 시스템에서 실행 불가능한 테스트 케이스를 생성할 수 있다. 또, 사용자에 의해 입력되는 대표 값은 테스팅 자동화를 위해 해결해야 할 향후 과제이다.

3. 요구사항 다이어그램

임베디드 시스템의 단일 요구사항을 하나의 모델로 1:1 대응 시키기 위한 도구로 REED(REquirement EDitor)라는 도구가 개발되었다. REED 는 그래픽 언어를 이용하여 요구사항 다이어그램을 표현하고 관리한다. 요구사항 다이어그램은 엔티티 블록(Entity Block)과 연산 블록(Operation Block) 그리고 이들 사이의 연결을 나타내는 링크로 구성된다[1]. 그림 1 는 시스템의 전원 스위치가 인가되면 5 초 동안 디스플레이에 "System Start" 라는 메시지가 출력되는 시스템의 요구사항을 표현한 다이어그램이다. 그림 1 의 요구사항 모델은 다음의 실제 요구사항에 대한 모델이다.

시스템은 SW 가 켜지면 5 초 동안 Display 에 'System Start'가 표시된다.



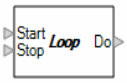
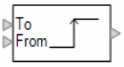
(그림 1) 단일 요구사항에 대한 다이어그램 표현

요구 사항 모델에 쓰인 블록들의 세부 기능은 표 1 에 나타나 있다. 표 1 에 나타난 블록들의 기능으로 그림 1 의 요구사항은 다음과 같이 동작한다.

먼저 SW 엔티티에 0v 가 입력 된 후 5v 가 입력 되면, Trigger 블록의 From 포트와 To 포트에 차례대로 활성화 신호가 인가되어 Trigger 블록이 동작한다. 다이어그램에서 블록이 동작을 하면 출력 포트를 통해 링크와 연결된 블록으로 활성화 신호가 나간다. Trigger 블록의 동작은 연결된 Timer 블록을 링크에 값에 따라, 0 초로 초기화 시키고, Loop 블록의 Start 포트에 활성화 신호를 인가하여 Loop 블록을 동작시킨다. 초기화된 Timer 블록은 시간의 흐름에 따라 경과된 시간을 출력하고, Loop 블록은 Stop 포트로부터 활성화 신호가 인가 될 때까지 Do 포트를 통해 활성화 신호를 계속 내보낸다. Loop 블록이 동작하는 동안 Loop 블록의 Do 포트와 연결된 Display 엔티티에는 'System Start' 라는 값이 출력된다. 시간이 흘러 Timer 블록이 5 초를 출력하면, Loop 블록의 Stop 포트에 활성화 신호가 인가되고, Loop 블록은 동작을 멈춘다.

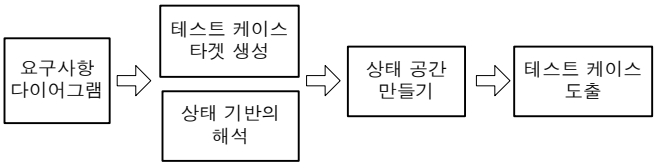
동작을 멈춘 Loop 블록은 Do 포트에 더 이상 활성화 신호를 내보내지 않으며, Display 엔티티에 출력 중이던 'System Start'값은 사라진다.

<표 1> 요구사항 다이어그램을 구성하는 각 블록의 설명

블록 표기	이름	설명
	엔티티 블록	요구사항 모델의 외부 입력과 출력을 나타낸다.
	Loop 블록	Start 포트로 활성화 신호가 인가되면 Stop 포트로 활성화 신호가 인가 될 때까지 Do 포트에 활성화 신호를 출력한다.
	Trigger 블록	From 포트에 활성화 신호가 인가된 후 To 포트로 활성화 신호가 인가되면 출력 포트로 활성화 신호를 출력한다.
	Do 블록	Do 블록이다. 입력 포트로 활성화 신호가 인가되면, 출력 포트로 활성화 신호를 내보낸다.
	Timer 엔티티 블록	시간이 입력되면 타이머는 동작하며, 흘러간 시간을 출력한다.

4. 테스트 케이스 생성 절차

그림 2 는 본 논문에서 제시하는 테스트 케이스 생성 절차를 나타낸다.



(그림 2) 테스트 케이스 생성 절차

4.1 테스트 케이스 타깃 생성

테스트 케이스 타깃은 요구사항 모델에 대해 구하고자 하는 테스트케이스의 조건이다. 요구사항 다이어그램에서 타깃의 구성은 선택한 커버리지와 다이어그램을 구성하는 블록들의 종류에 따라 달라진다. 그림 1 의 모델에 대해 테스트 케이스를 구하기 위해서는 표 2 와 같은 타깃 정보가 설정된다. 표 2 는 그림 1 요구사항 다이어그램에서 Loop 블록을 대상으로 MC/DC 를 위해 생성된 타깃의 정보이다. 전체 테스트 케이스 타깃은 여러 개의 개별 타깃으로 구성된다. 하나의 개별 타깃은 하나의 테스트 케이스에 대한 조건을 명세 한다. 조건은 여러 개의 실행 단계로 이루어 질 수 있다. 하나의 실행 단계는 해당 단계에서 만족해야 할 포트들의 입력 신호에 대한 조건들을 나

타낸다. 표 2 는 전체 2 개의 타깃을 나타내며, 타깃 No.1 의 내용은 실행단계 1 에서 Loop 블록의 입력으로 Start 포트가 활성화 된 후 Do 포트가 활성화 되고, 실행단계 2 에서 Stop 포트의 활성화로 Do 포트로 비활성화 신호가 나와야 된다는 조건을 나타내고 있다. 표 2 의 전체 타깃의 내용을 만족하는 테스트 케이스는 Loop 블록에서 Stop 포트에 대한 독립적인 영향을 보일 수 있다. 따라서 타깃 No.1 과 No.2 가 모두 만족된다면 Loop 블록에 대해 MC/DC 를 만족하는 테스트 케이스가 완성 된다. 요구사항 다이어그램을 구성하는 블록 전체에 대해 이와 같이 MC/DC 에 대한 타깃이 설정되고, 타깃을 만족하는 테스트 케이스를 구한 다면, 결과로 얻어진 테스트 케이스는 MC/DC 를 만족한다.

<표 1> Loop 블록에 대한 테스트 케이스 타깃

타깃 No.	블록 이름	실행 단계	포트 이름	활성화/비활성화 신호 A: 활성화, D: 비활성화
1	Loop	1	Start	A
			Stop	
			Do	A
		2	Start	
			Stop	A
			Do	D
2	Loop	1	Start	A
			Stop	
			Do	A
		2	Start	
			Stop	D
			Do	A

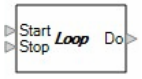
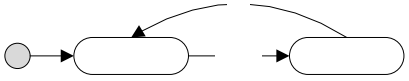
4.2 상태 기반의 해석

2 장의 관련연구에서 살펴 본 바와 같이, 상태 기반의 요구사항 모델에 대해서 테스트 케이스 생성 기법은 존재한다. 그래픽 언어 기반의 요구사항 다이어그램에 대해 테스트 케이스 생성은 이와 같은 기법을 바로 적용하기가 어렵다. 명시적으로 상태 기반의 정보를 포함하고 있지 않은 그래픽 언어 기반의 요구사항 모델에 대해, 본 논문에서 제시하는 방법은 요구사항 다이어그램 모델을 상태 기반으로 해석하고, 모델의 실행을 바탕으로 테스트 케이스를 생성하는 기법이다. 그래픽 언어 기반의 요구사항 모델을 상태 기반으로 해석할 수 있다면, 앞서 정의한 테스트 케이스 타깃과, 모델의 실행을 바탕으로 모델의 실행 트리를 구성할 수 있다. 생성된 테스트 케이스는 모델의 실행을 바탕으로 얻어졌기 때문에, 실제 시스템에 항상 입력 가능하며, 테스트 오라클을 함께 제공한다.

요구사항 다이어그램의 상태 기반 해석을 위해서는 표 3 과 같이 다이어그램을 구성하는 블록들에 대한 상태 기반 정보가 필요하다.

표 3 은 그림 1 의 요구사항 다이어그램을 구성하는 블록들 중에서 연산 블록에 해당하는 것들의 상태 기반 해석을 나타내고 있다. 표 1 에 나타나 있는 Loop 블록의 행동은 표 3 의 상태 기반 해석으로도 설명할 수 있다. 표 3 에 의하면 Loop 블록은 내부적으로 최초 S0 에 머무르면서, Start 포트의 입력으로 활성화가 인가되면, 내부적으로 S1 의 상태로 천이를 한다. 그리고, S1 의 상태에서는 Do 블록으로 활성화 신호를 출력하고, Stop 포트에 활성화 신호가 인가되면 다시 S0 의 상태로 천이하며 Do 포트로 비활성화 신호를 출력한다.

<표 2> 각 블록들의 상태 정보

블록	상태 기반 해석
	
	

4.3 요구사항 모델의 상태 공간 만들기

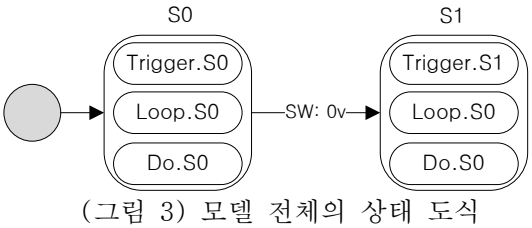
지금까지 요구사항 다이어그램을 상태 기반으로 해석하기 위해 전체 요구사항 모델을 구성하는 개별 블록의 상태 기반 정보를 알아 보았다. 요구사항 모델의 전체 상태는 앞서 정의한 모델을 구성하는 개별 블록의 상태의 조합으로 정의한다. 요구사항 모델의 상태를 표현하고 구분할 수 있는 규칙이 마련되었다면, 테스트 케이스를 찾기 위해 모델의 상태 공간 (state space)을 만들어야 한다. 상태 공간을 만들기 위한 알고리즘은 표 4 와 같다.

<표 3> 상태 공간 구축의 알고리즘

- ✓ stateSpace : 상태공간
- ✓ startPoint: 상태공간 상의 임의의 시작점
- ✓ newState: 도달 가능한 새로운 상태

```
Loop (stateState conformed all Target)
        startPoint = StateSpace.getState()
          find = startPoint.findNew(newState)
        If( find)
            stateSpace.addState(newState)
End Loop
```

상태 공간은 최초 모델의 상태로부터 입력에 대해 모델이 도달 가능한 상태들로 구성된 트리의 형태를 갖는다. 그림 3 은 그림 1 요구사항의 전체 상태를 도식화 하고, 상태를 바탕으로 트리를 구성해 나가는 모습의 일부분이다.

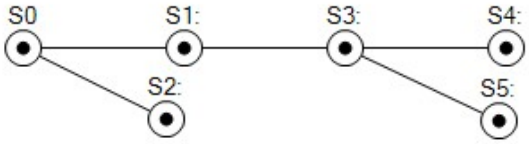


(그림 3) 모델 전체의 상태 도식

표 4 에 나타난 알고리즘의 표기로 그림 3 을 해석하면, 그림 3 은 startPoint 로 S0 가 선택된 후, S1 이 newState 로 발견되어 전체 상태 공간에 추가된 트리의 모습이다. 이 때 S0 에서 S1 으로 이르게 하는 모델의 입력 값은 트리의 링크를 나타내며, 이 경우는 SW 엔티티로 입력한 0v 이다. 알고리즘의 루프는 주어진 테스트 케이스 타깃이 만족될 때까지 진행 되며 루프가 진행되는 동안 모델의 상태 공간을 나타내는 실행 트리는 자라게 된다.

4.4 테스트 케이스 도출

그림 4 는 그림 1 의 요구사항에 대해 구축한 상태 공간의 전체 모습이다.



(그림 4) 그림 1의 요구사항 모델의 상태 공간

그림 4 에서 초기 상태에서 S5 로 가는 입력 값에 따른 모델의 상태 변화는 표 5 와 같다.

<표 4> 모델이 s5 로 가기 위한 입력 값에 따른 모델의 상태변화 정보

단계	입력 값	모델의 상태 변화	
		시작 상태	도착 상태
1	SW: 0v	S0	S1
2	SW: 5v	S1	S3
3	5 초 대기	S3	S5

그림 4 에 나타난 상태 S5 가 표 2 의 테스트 케이스 타깃 No.1 을 만족할 때, 표 5 에 나타난 입력 값들의 단계가 테스트 케이스 타깃 NO.1 을 만족하는 테스트 케이스가 된다.

본 논문에서 제시하는 기법에서 테스트 케이스 도출은 앞서 구한 모델의 상태 공간에서 개별 테스트 케이스 타깃을 만족하는 상태들을 찾는 과정이다. 테스트 케이스 타깃을 만족하는 상태는, 표 5 와 같이 초기 상태로부터 해당 상태로 이르는 경로의 입력 값들을 조합하여 테스트 케이스로 이용할 수 있다. 모델의 상태 공간을 구축하기 전, 테스트 케이스 타깃이 MC/DC 를 만족하는 조건들로 설정 되었다면, 도출되는 테스트 케이스는 MC/DC 를 만족하는 테스트 케이스가 된다.

5. 결론 및 향후 과제

지금까지, 단일 요구사항의 표현에 적합한 그래픽 언어 기반의 요구사항 다이어그램으로부터 테스트 케이스를 생성하기 위한 다이어그램의 해석 및 테스트 케이스 생성 기법에 대해 알아 보았다. 기존의 테스트 케이스 생성을 위한 많은 연구들은 상태 기반의 요구사항 모델에 대해 적용 가능한 기법들로, 그래픽 언어 기반의 요구사항에 대해 바로 적용할 수 없다. 본 논문에서 제시한 요구사항 모델에 대한 상태 기반의 해석 기법은 명시적으로 상태 정보를 포함하지 않은 그래픽 언어 기반의 요구사항 모델에 대해 상태 기반의 테스트 케이스 생성 기법을 적용할 수 있다. 이 기법을 통해 생성된 테스트 케이스는 테스트 케이스 타깃과 모델의 실행을 바탕으로 생성되어, 다이어그램 모델의 정적인 분석을 통해 생성된 테스트 케이스와 달리, 항상 실행 가능한 형태이다.

현재 상태 공간의 실행 트리를 구축 하는 과정은 입력 값의 임의 생성으로 이루어 지고 있다. 테스트 케이스 타깃을 만족하는 모델의 상태를 빠르게 찾기 위해서는, 주어진 요구사항 모델의 상태 공간을 효율적으로 구축하는 연구가 진행되어야 한다.

참고문헌

[1] 오정섭, 이홍석, 박현상, 김장복, 최경희, 정기현, “그래픽 언어를 이용한 임베디드 시스템의 단일 요구사항 모델링”, 한국정보처리학회논문지 D, v.15D, no.4, pp.505-512, 2008. 8

[2] 원종섭, 최경희, 정기현, “도형 요구사항의 MC/DC 테스트 케이스 생성 기법”, 한국정보처리학회 춘계학술발표대회 논문집 제 15 권 제 1 호, 2008. 5

[3] S. Rayadurgam and M. P. E. Heimdahl. Test-sequence generation from formal requirement models. In HASE ’01: The 6th IEEE International Symposium on High-Assurance Systems Engineering, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0769512`55.

[4] Telelogic AB, http://www.telelogic.com/products/doors/doors/index.cfm

[5] Borland Software Co., http://www.borland.com/us/products/caliber/index.html

[6] Compuware Co., http://www.compuware.com/products/optimaltrace/

[7] Sanjai Rayadurgam and Mats P. E. Heimdahl. Coverage Based Test-Case Generation Using Model Checkers. In Proceedings of the 8th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2001), pages 83–91, Washington, DC, April 2001. IEEE Computer Society.

[8] James Bach, “Risk and Requirements-Based Testing,” IEEE Computer, Vol.32, No.6, pp.113-114, June, 1999.

[9] Jeffrey M. Thompson, Mats P. E. Heimdahl, and Steven P. Miller. Specification-based prototyping for embedded systems. In ESEC/FSE-7: Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering, pages 163–179, London, UK, 1999. Springer-Verlag. ISBN 3-540-66538-2. doi: 10.1145/318773.318940.

[10] John Callahan, Francis Schneider, and Steve Easterbrook. Automated Software Testing UsingModel-Checking. In *Proceedings 1996 SPIN Workshop*, August 1996. AlsoWVU Technical Report NASAIVV-96-022.