Journal of Institute of Control, Robotics and Systems (2010) 16(8): 786-793
DOI 10.5302/J.ICROS.2010.16.8.786 ISSN 1976-5622

로봇용 소프트웨어 컴포넌트에서 비정상 천이를 포함한 상태 천이 시퀀스용 테스트 스윗 생성 기법

Generation Algorithm of Test Suite for State Transition Sequence with Abnormal Transitions in Robot Software Component

맹 상 우, 박 홍 성*
(Sang-Woo Maeng[1] and Hong Seong Park[1])
[1]Kangwon National University

Abstract: This paper proposes a new method called the path-history coverage to generate a test suite to test the state transition behavior of the robot SW component. The proposed method generate a test suite which includes abnormal state transitions based on FSM of target component. Especially the proposed method covers the disadvantage of the mutation test method that the size of the test suite is explosively increasing. Examples including OPRoS Component[1] show the validity of the proposed method.

Keywords: component state test, abnormal transition, path-history coverage

I. 서론

로봇 소프트웨어 플랫폼인 OPRoS (Open Platform for Robotic Services)[2]는 소프트웨어 재활용성과 쉬운 사용을 위하여 컴포넌트 기반으로 개발되었다. 이러한 소프트웨어 컴포넌트를 이용한 개발은 모듈의 재사용성을 통해 다양한 응용프로그램을 신속하게 개발할 수 있도록 하여 생산성을 높여주는 효율적인 방법이다[3]. 그러나 개발된 응용프로그램의 안정성과 신뢰성을 보장하기 위해서는 개개의 컴포넌트를 면밀히 테스트해야 한다. 따라서 컴포넌트의 재사용성의 측면을 고려한 테스트 방법의 연구가 요구되며 이러한 연구의 일환으로 컴포넌트의 상태 천이 테스트 기법이 연구되어왔다[4-7].

정밀한 테스트를 위하여 일반적인 상태 천이 테스트 기법에 변이(mutation) 기법을 적용시킨 연구도 있었다[8,9]. 변이 테스트 방법은 변이 연산자를 정의하여 본래 컴포넌트의 상태도를 변형시킨 상태도를 생성한 후, 이 둘을 차별화 시킬 수 있는 일련의 테스트 스윗(suite)을 이용하여 에러를 검출해 낸다. 그러나 변이 테스트 기법은 테스트 스윗을 구성하는 테스트 케이스의 수가 지나치게 많이 생성될 수 있고 테스터의 노력이 많이 들어가는 단점이 있다. 따라서 정밀하고 효율적인 테스트를 위하여 작업 과정을 자동화시키고 테스트 수행 시간을 줄일 수 있는 연구가 필요하다.

본 논문에서는 컴포넌트의 상태 천이를 시험하기 위한 새로운 테스트 스윗 생성 기법을 제안하기 위하여 [10]을 확장한다. 여기서 테스트 스윗은 테스트 케이스 등의 집합을 의미한다. 제안하는 방법에 의해 생성된 테스트 스윗은 컴포넌트의 각 상태에서 유효하지 않은 상태천이를 일으키게 한다. 이러한 테스트 스윗은 변이 테스트 기법에 사용되기 적합하다. 그리고 테스트 케이스를 생성하기 위하여 정형화된 유한상태기계(FSM: Finite State Machine) 명세를 이용하므로 특정 언어나 컴포넌트 모델에 대해 독립적이다. 또한 생성된 테스트 스윗은 테스트 대상 컴포넌트의 소스코드를 변형시키지 않고도 테스트에 적용할 수 있다. 따라서 상용 컴포넌트와 같이 실행 코드만 제공되는 컴포넌트에 적용이 가능하므로 실제 테스트에 적용하기 용이하다.

일반적인 상태테스트의 수행시간은 크게 테스트 스윗을 구성하는 테스트 케이스의 개수와 테스트 케이스를 구성하는 천이의 개수에 의해 결정될 수 있다. 제안하는 방법은 테스트 케이스를 구성하는 비정상 천이 경로의 수를 줄이고 감소 정도를 조절할 수 있는 방법을 포함한다.

본 논문의 구성은 다음과 같다. II 장에서는 상태 변이 테스트에 관한 기존의 연구를 살펴본다. III 장에서는 제안하는 테스트 스윗 생성 방법을 설명한다. 또한 테스트의 수행시간을 줄이기 위해 테스트 케이스에 포함된 비정상적인 상태 천이의 발생 횟수를 합리적으로 줄이는 방법을 설명한다. IV 장에서는 OPRoS 컴포넌트를 대상으로 실제 생성된 테스트 스윗을 적용하여 실험한 결과를 기술한다. 마지막으로 V 장에서 결론을 맺는다.

II. 관련연구

거의 모든 컴포넌트의 FSM은 순환(loop) 구조를 포함하고 있으므로 컴포넌트 테스트를 위한 상태천이 함수의 호출 순서와 천이 후 상태 쌍은 무한할 수 있다[4]. 이러한 문제를 해결함과 동시에 상태천이 함수에 대해 적합한 커버리지 기준을 세우고 유한한 상태 천이 시퀀스를 생성하는 여러

* 책임저자(Corresponding Author)
논문접수: 2010. 4. 16., 수정: 2010. 6. 4., 채택확정: 2010. 6. 10.
맹상우, 박홍성: 강원대학교 전자통신공학과
(mswok@control.kangwon.ac.kr/hspark@kangwon.ac.kr)
※ 본 논문은 2010년도 ICROS 학술대회에서 초안이 발표되었고, 지식경제부(M.K.E)가 지원하였다.

Copyright© ICROS 2010

기존의 연구가 있었다[4-7]. [5]에서는 오토마타를 대상으로 하여 입력 순서에 의한 상태천이 테스트 방법을 연구하였다. 또한 상태테스트에서 상태도에 대한 변이가 생기고 이를 차별화시키는 개념을 처음 소개하였다. [6]과 [7]에서는 일반적인 상태 천이 경로를 순회(traverse)하는 테스트 스윗을 생성하기 위해 새로운 커버리지 개념을 제안하였다.

이들 방법과는 다르게 [8]에서는 클래스를 테스트하기 위해 상태 기반 변이연산자를 제안하고 테스트 케이스의 생성기법을 제시 하였다. 그리고 [9]에서는 변이 테스트 기법을 기반으로 FSM을 검증하기 위해 9가지의 변이 연산자를 정의하고 테스트 도구를 구현하였다.

이러한 연구들[8,9]에서 제안하는 변이 연산자들은 주로 상태와 천이 함수, 출력결과에 대한 교체와 삭제로 이루어져 있다. 그러나 해당 변이 연산자를 이용해 변이 시킨 상태도를 원래의 상태도와 차별화 시킬 수 있는 테스트 스윗을 자동으로 생성시키기 어렵다. 또한 일부 변이 연산자의 경우, 소스코드가 제공되지 않는 컴포넌트에 대해서는 실제 테스트 수행시 제약이 따른다. 또한 변이 연산자에 의해 모든 변이를 생성하게 되면 FSM의 복잡도에 따라 그 수가 매우 커질 수 있으므로 테스트 수행시 많은 시간과 노력을 필요로 한다는 단점이 있다.

따라서 이 논문에서는 테스트 수행에 걸리는 시간을 줄이기 위하여 자동화가 가능한 비정상적인 천이 발생을 포함하는 테스트 스윗 생성 방법을 제안한다. 그리고 비정상적인 상태 천이의 발생 횟수를 합리적으로 줄이기 위한 새로운 커버리지를 함께 제안한다.

III. 테스트 스윗 생성

이 절에서는 기존의 상태 테스트 커버리지 기준에 의해 생성된 테스트 스윗에 비정상적 천이 호출을 포함시키는 방법을 제안한다. 컴포넌트의 상태는 FSM을 이용하여 표현될 수 있다. 이 논문에서 다루는 FSM은 초기상태가 하나 이상의 경우도 표현할 수 있어야 하기 때문에 기존의 결정론적 FSM 모델을 확장하여 다음과 같이 7개의 요소로 새롭게 정의 한다.

- S : 1개 이상의 유한 상태 집합
- S_O : 1개 이상의 초기 상태 집합, $S_O \subset S$
- F : 종료 상태 집합, $F \subset S$
- $\sum$: 1개 이상의 입력 기호 집합
- δ : 상태 천이. $\delta : S \times \sum \rightarrow S$ 로 정의된다. 예를 들면, 상태 s_1에서 입력 t가 발생하여 s_2로 천이할 경우 $s_1 \xrightarrow{t} s_2$ 로 표기할 수 있음.
- s_v : 가상 초기 상태. FSM에 S_O 의 원소가 1개 이상 존재할 경우 유일한 시작점을 대표하기 위해 도입함. $s_v \in S$
- Σ_I : s_v를 도입함에 따라 추가되는 s_v에서 각 초기상태 s_o로 향하게 하는 입력기호의 집합. $\Sigma_I \subset \sum$

위의 요소를 이용하여 $FSM = (S, S_O, F, \sum, \delta, s_v, \Sigma_I)$로 정의할 수 있다. 일반적으로 상태테스트 케이스는 각각 한 개씩의 전 상태(pre-state), 천이, 후 상태(post-state)의 순서쌍으로 구성된다. 정상적인 상태 천이란 해당 컴포넌트 모델의 FSM의 정의를 따라 특정 상태에서 일어날 수 있는 이벤트가 발생하거나, 혹은 호출 가능한 천이 함수를 수행하는 경우를 일컫는다. 반면 비정상적인 상태 천이란 특정 상태에서 일어날 수 없는 이벤트가 발생하는 경우나 호출 될 수 없는 천이 함수를 수행하는 경우를 의미한다.

본 논문에서 제안하는 테스트 스윗 생성 기법은 FSM의 각 상태에 대해 전체 천이 함수를 호출하여 비정상적인 상태 천이 호출을 테스트 스윗에 추가한다. 즉 $S \times \sum = \{(s,t) | s \in S, t \in \sum\}$를 만족하는 모든 천이를 일으킬 수 있도록 테스트 스윗을 생성한다.

이러한 테스트 스윗은 기존의 소프트웨어 모듈을 이용한 개발 시, 소스가 제공되지 않더라도 상태 모델이 제공되면 적용이 가능하다. 또한 컴포넌트의 프레임웍이 컴포넌트 상태에 대한 선조건(pre-condition)을 검사하지 않고 천이 함수를 호출시켜 발생하는 에러를 검출하기 적합하다.

먼저 FSM으로부터 테스트 스윗을 생성하기 위해서는 상태 테스팅 트리(state testing tree[5])를 생성하여야 하기 때문에 해당 FSM 모델의 시작상태는 반드시 하나로 지정되어야 한다. 그러나 실제 컴포넌트의 경우, 구현에 따라 여러 개의 생성자에 의해 시작 상태가 한 개 이상이 존재할 수 있다. 따라서 어떠한 경우에도 FSM의 시작상태를 하나로 일관되게 유지하기 위하여 가상초기상태 s_v를 FSM에 도입한다. s_v를 도입함에 따라 s_v로부터 본래의 컴포넌트 FSM에서 정의하는 한 개, 혹은 그 이상의 시작상태로 향하는 천이 집합 Σ_I를 추가해야 한다. 그리고 컴포넌트가 파괴되어 더 이상 동작할 수 없는 상태를 처리하기 위하여 종료상태 F라는 특정 속성을 가진 상태를 새롭게 추가한다. 테스트 스윗의 생성 과정은 크게 다음의 세단계로 구성된다.

1) 만약 테스트 대상 컴포넌트의 FSM 모델에 두 개 이상의 초기상태가 존재한다면 s_v와 Σ_I를 추가하여 초기화 한다.
2) FSM으로부터 기존의 상태 테스트 커버리지 기준을 선택하여 유한한 상태천이 순서를 나타내는 상태 테스팅 트리를 생성한다.
3) 상태 테스팅 트리의 각 단말노드에 이르는 경로를 추출하여 비정상 천이를 추가하고 테스트 스윗을 생성한다.

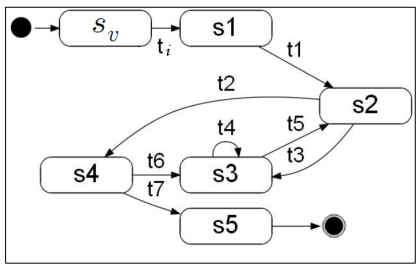


그림 1. s_v와 Σ_I가 추가된 FSM 모델.

Fig. 1. FSM model which includes s_v and Σ_I.

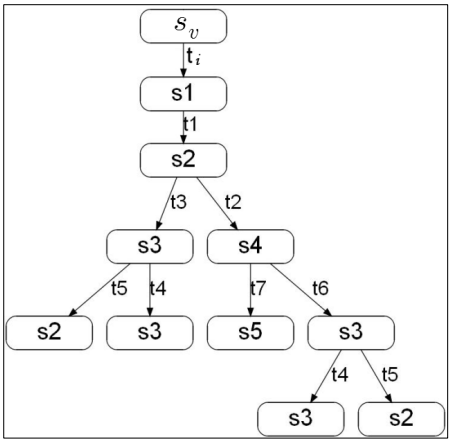


그림 2. 왕복 커버리지를 이용해 생성된 상태 테스팅 트리.
Fig. 2. State testing tree generated by round-trip coverage.

그림 1과 그림 2에서 둥근 사각형은 상태를, 선은 천이를 표시한다. 그림 1의 예제는 첫 번째 단계인 FSM의 초기화 작업을 마친 모습이다. 본래 FSM에 정의된 초기 상태와 종료 상태 집합은 각각 $S_O=\{s1\}$, $F=\{s5\}$와 같다. 초기화 과정에서 하여 s_v가 추가 되었고 S_O에 한 개의 상태가 있으므로 $\Sigma_I=\{t_i\}$가 더해진다.

그림 2는 그림 1의 FSM에 왕복(round-trip) 커버리지 기준[7]을 적용해 추출한 상태 테스팅 트리이다. 왕복 커버리지 방식은 모든 천이 커버리지(all-transition coverage), 천이쌍 커버리지(transition-pair coverage)와 전체서술어 커버리지(full-predicate coverage)를 모두 커버할 수 있는 적합한 커버리지 기준이다[6]. 예제에서는 같은 노드를 두 번 만났을 때뿐만 아니라 종료 상태인 S_F에 도달했을 때에도 상태 테스팅 트리의 생성을 마치도록 하였다. 다음으로 테스트 스윗에 비정상 천이 호출을 추가하는 방법을 설명하기 위해 아래의 기호를 정의한다.

- $M(s)$: $s\in S$, $t\in\sum$일 때 $s\xrightarrow{t}s$ 로 일어날 수 없는 입력 기호의 집합. 즉 FSM의 s에서 호출될 수 없는 천이의 집합.
- $s_r\in S$: 천이가 발생되기 이 전의 상태 (pre-state)
- $s_o\in S$: 천이가 발생된 후의 상태 (post-state)
- E_N : 정상 천이시 전 상태, 정상 천이를 위한 입력기호와 후 상태의 3개의 원소로 구성된 순서쌍이며 다음의 집합으로 표시될 수 있다.

$$E_N=\left\{(s_i,t,s_j)|s_i,s_j\in S,t\in\sum,s_i\xrightarrow{t}s_j\right\}$$

- E_A : 비정상 천이 시 전 상태, 비정상 천이를 일으키는 입력기호와 후 상태의 3개의 원소로 구성된 순서쌍이며 다음의 집합으로 표시될 수 있다.

$$E_A=\left\{(s_i,t,s_i)|s_i\in S,t\in M(s_i),s_i\xrightarrow{t}s_i\right\}$$

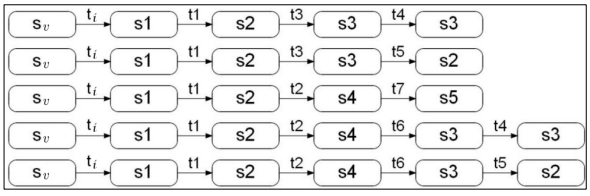
- $S_{TC}=\{e_1,e_2,e_3\cdots e_i|e_i\in E_N\vee e_i\in E_A\}$: 테스트 케이스
- $S_{TS}=\{S_{TC}\}$: 테스트 스윗. 하나의 FSM을 정해진 커버리지 기준을 만족 시키도록 테스트하기 위한 일련의 테스트 케이스 집합
- $IPG(STT)$: IPG (Independence Path Graph). 상태 테스팅 트리 STT (State Testing Tree)가 주어질 때 트리의 뿌리노드부터 단말노드에 이르는 각각의 경로를 연결 리스트 형태로 구성시킨 그래프

$M(s)$는 s에서 호출될 수 없는 비정상 적인 천이의 집합이다. 예를 들어 그림 1의 상태도에서 $s2$의 경우 $t2$와 $t3$만 호출이 가능하다. 따라서 FSM의 전체 입력기호 $\sum$ 중 $t2$,$t3$를 제외한 나머지 입력기호로 구성되며 $M(s1)=\{t1,t4,t5,t6,t7\}$ 와 같이 표현할 수 있다.

s_r과 s_o는 특정 상태가 아닌 테스트 진행 과정에 있어 현재의 상태와 천이함수가 호출되고 난 후의 상태를 상대적으로 표현한다. E_N과 E_A 역시 천이함수의 종류가 아닌 전 상태(pre-state)에서 호출 가능한 천이함수인가의 여부에 따라 구분된다. 테스트 케이스는 위에서 표현한 E_N과 E_A의 순서 있는 조합으로 정의된다.

IPG는 상태 테스팅 트리로부터 생성된다. 시작상태인 뿌리노드부터 각 단말노드에 이르는 경로를 하나씩 분리하여 하나의 단방향 연결 리스트의 형태로 떼어낸다. IPG는 이러한 경로의 집합으로 구성되는 그래프 구조이다. 그림 2의 상태 테스팅 트리로부터 일반적인 방법으로 생성한 E_N으로 구성된 테스트 스윗과 IPG는 그림 3에 표시하였다.

정상적인 상태 천이로 구성된 테스트 스윗에 오류를 삽입하기 위하여 IPG의 각 상태에서 모든 비정상 천이를 유발시킨다면 FSM의 상태 및 천이 개수에 따라 테스트 스윗을 구성하는 테스트 케이스의 천이 경로가 매우 길어질 수 있다. 천이 경로의 길이는 테스트의 수행시간을 결정짓는 중요한 요소이다. 따라서 비정상 천이함수 호출의 삽입 횟수를 줄여 테스트 수행시간을 감소시키기 위한 방법으로 경로 이력 커버리지(path-history coverage) 방법을 제안한다. 이 커버리지 기준은 IPG의 각 경로를 순회할 때 현 상태에



(a) IPG

$$S_{TC_1}=\{(s_v,t_i,s1),(s1,t1,s2),(s2,t2,s4),(s4,t7,s5)\}$$
$$S_{TC_2}=\{(s_v,t_i,s1),(s2,t2,s4),(s4,t6,s3),(s3,t4,s3)\}$$
$$S_{TC_3}=\{(s_v,t_i,s1),(s2,t2,s4),(s4,t6,s3),(s3,t5,s2)\}$$
$$S_{TC_4}=\{(s_v,t_i,s1),(s2,t3,s3),(s3,t4,s3)\}$$
$$S_{TC_5}=\{(s_v,t_i,s1),(s2,t3,s3),(s3,t5,s2)\}$$
$$S_{TS}=\{S_{TC_1},S_{TC_2},S_{TC_3},S_{TC_4},S_{TC_5}\}$$

(b) Test suite

그림 3. 그림 2의 상태 테스팅 트리로부터 생성한 IPG와 테스트 스윗.
Fig. 3. IPG and test suite generated from state testing tree in Fig. 2.

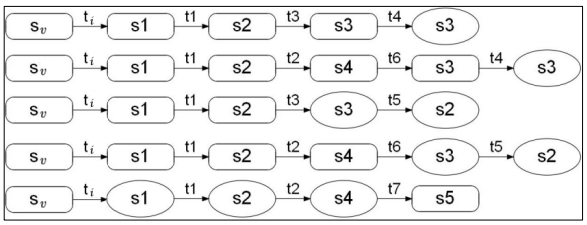
이르기까지 거쳐 온 경로 이력을 식별자로 하여 각 상태에서 새로운 경로 이력을 가질 때에만 비정상 천이를 호출시키도록 결정한다. 경로 이력을 식별자로써 활용하여 테스트 스윗을 생성하는 방법을 설명하기 위해 다음의 기호를 정의한다.

- $P=\{(t,s)\,|\,t\in\delta, s\in S\}$: 천이에 해당되는 입력 기호와 천이가 된 후 상태로 구성된 순서쌍의 집합
- $U=\{p_1,p_2,p_3\ldots p_n\,|\,p_i\in P, i=1\ldots n\}$: 단위 천이 시퀀스(unit transition sequence, UTS)의 집합. 최대 n개의 P를 포함할 수 있는 순서쌍의 집합. 여기서 n은 UTS의 크기(혹은 |U|) 를 의미한다.

UTS는 상태 테스팅 트리의 각 경로를 루트 노드부터 순회할 때 특정 시점의 상태까지 거쳐 온 경로를 구분하기 위한 경로 이력 정보를 표현하는 집합이다. P는 UTS를 표현하기 위한 기본 단위가 된다. 그림 3의 IPG에서 s_v, $s1$, $s2$, $s4$, $s3$, $s2$를 거치는 경로를 순회할 때, 특정 시점에서 머무르는 상태 s_{cur}에서 UTS의 크기 n에 따른 UTS 값은 표 1과 같다.

표 1. UTS의 크기가 2, 3일 때, 각 상태에 대한 따른 UTS 값.
Table 1. UTS values for each state when UTS size is 2,3.

s_{cur}	n = 2	n = 3
s_v	$U=\Phi$	$U=\Phi$
$s1$	$U=\{(t_i,s1)\}$	$U=\{(t_i,s1)\}$
$s2$	$U=\{(t_i,s1),(t1,s2)\}$	$U=\{(t_i,s1),(t1,s2)\}$
$s4$	$U=\{(t1,s2),(t2,s4)\}$	$U=\{(t_i,s1),(t1,s2),(t2,s4)\}$
$s3$	$U=\{(t2,s4),(t6,s3)\}$	$U=\{(t1,s2),(t2,s4),(t6,s3)\}$
$s2$	$U=\{(t6,s3),(t5,s2)\}$	$U=\{(t2,s4),(t6,s3),(t5,s2)\}$



(a) IPG

$$TC_1=\left\{\begin{array}{l}(s_v,t_i,s1),(s1,t1,s1),(s1,t4,s1),(s1,t5,s1),\\(s1,t6,s1),(s1,t7,s1),(s1,t1,s2),(s2,t3,s2),\\(s2,t2,s2),(s2,t1,s2),(s2,t4,s2),(s2,t5,s2),\\(s2,t2,s4),(s4,t7,s5)\end{array}\right\}$$

$$TC_2=\left\{\begin{array}{l}(s_v,t_i,s1),(s1,t1,s2),(s2,t2,s4),(s4,t3,s4),\\(s4,t2,s4),(s4,t1,s4),(s4,t6,s4),(s4,t7,s4),\\(s4,t6,s3),(s3,t3,s3),(s3,t2,s3),(s3,t1,s3),\\(s3,t6,s3),(s3,t7,s3),(s3,t4,s3)\end{array}\right\}$$

$$TC_3=\left\{\begin{array}{l}(s_v,t_i,s1),(s1,t1,s2),(s2,t2,s4),(s4,t6,s3),\\(s3,t1,s3),(s3,t4,s3),(s3,t5,s3),(s3,t6,s3),\\(s3,t7,s3),(s3,t5,s2)\end{array}\right\}$$

$$TC_4=\left\{\begin{array}{l}(s_v,t_i,s1),(s1,t1,s2),(s2,t3,s2),(s2,t2,s2),\\(s2,t1,s2),(s2,t6,s2),(s2,t7,s2),(s2,t3,s3),\\(s3,t3,s3),(s3,t2,s3),(s3,t1,s3),(s3,t6,s3),\\(s3,t7,s3),(s3,t4,s3)\end{array}\right\}$$

$$TC_5=\left\{\begin{array}{l}(s_v,t_i,s1),(s1,t1,s2),(s2,t3,s3),(s3,t1,s3),\\(s3,t4,s3),(s3,t5,s3),(s3,t6,s3),(s3,t7,s3),\\(s3,t5,s2)\end{array}\right\}$$

(b) Test suite

그림 4. 비정상 천이함수를 호출할 상태를 표시한 IPG 와 IPG로부터 생성한 테스트 스윗.

Fig. 4. IPG which state marked with abnormal transition call and test suite generated from IPG.

비정상 천이함수 호출을 하는 노드를 결정하기 위해 IPG의 각 경로를 순회하며 각 노드에 들렀을 때 해당하는 UTS 집합에 등록한다. 만약 현재 시점의 UTS를 등록할 때 동일한 값의 UTS가 해시집합에 존재하지 않는다면 해당 노드에서 $M(s)$를 호출시키도록 설정한다. 즉 동일한 경로 이력을 가진 상태의 경우에는 비정상 천이함수를 호출하지 않도록 한다. 테스터(혹은 테스트 수행자)는 FSM의 복잡성이나 특성에 맞추어 UTS의 크기를 이용하여 테스트 케이스의 천이경로 길이를 조절할 수 있다. 그림 4(a)는 경로 이력 커버리지 기준을 적용하여 비정상 천이함수를 호출하는 노드의 개수를 줄인 IPG로, 타원형으로 표시된 노드에서만 $M(s)$를 호출하도록 설정되어 있다.

비정상 천이함수가 호출될 노드가 결정되면 IPG로부터 테스트 스윗을 생성한다. 그림 4(b)는 그림 4(a)로부터 생성된 테스트 스윗이다. IPG의 각 경로 수는 테스트 케이스의 수와 일치한다. 즉 테스팅 트리의 단말 노드의 개수만큼 테스트 케이스가 생성될 수 있다. 테스트 스윗을 생성하기위해 IPG의 각 경로를 순회하는 중 비정상 천이를 호출해야 하는 상태를 만났을 경우, 그 상태에 해당하는 M(s)을 모두 호출 시키도록 하고 후 상태(post-state)는 자기 자신으로 지정한다. FSM으로부터 테스트 스윗이 생성되기까지의 과정의 순서도를 그림 5에 표시하였다.

그림 5의 순서도에 표시된 각 과정에 따른 설명은 다음과 같다.

(a) 정형화된 FSM 정보를 읽어 들여 s_v와 Σ_I를 추가하여 초기화시킨다. 여기서 FSM 정보는 XML로 정의되며, XML 스키마가 그림 6에 정의되어 있음.

(b) FSM으로부터 상태 테스팅 트리 STT를 생성한다. 각 상태에 대한 $M(s)$ 정보를 테이블로 저장한 최소 스패닝 트리를 생성한다.

(c) STT를 이용하여 IPG를 생성한다.

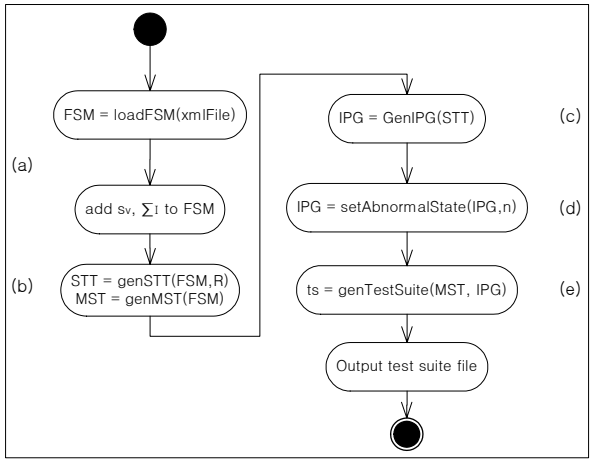


그림 5. 테스트 스윗 생성 과정 흐름도.

Fig. 5. Test suite generation flow chart.

(d) IPG를 순회하며 비정상 천이 함수를 호출시킬 노드를 지정한다.

(e) IPG를 토대로 테스트 스윗을 생성한다.

이 논문에서 제안하는 경로 이력 커버리지를 만족시키기 위해 비정상 천이 함수를 호출하는 상태를 설정하는 단계 (d)와 IPG로부터 테스트 스윗을 생성하는 단계 (e)에 대한 알고리즘을 그림 7과 그림 8에 각각 표시하였다.

```
<xs:schema>
  <xs:element name="FSM">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="stateList"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="stateList">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="state" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="state">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="transitionList" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="type">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="FINAL"/>
            <xs:enumeration value="INIT"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="name" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="transitionList">
   <xs:complexType>
     <xs:sequence>
     <xs:element ref="transition" maxOccurs="unbounded"/>
     </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="transition">
    <xs:complexType>
      <xs:attribute name="trigger" use="required"/>
      <xs:attribute name="goto" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

그림 6. FSM 모델 표현을 위한 XML 스키마.

Fig. 6. XML Schema for FSM model description.

```
IPG = setAbnormalState(IPG, n)

input     IPG : Indepent path graph
            n   : size of UTS
output    IPG : includes abnormal state
begin
    UTSHashSet = {∅}
    pQueue = {∅} // queue size = n,
                    means UTS for each state
    for each path in IPG
        clear pQueue
        for each node in path
            t = current transition
            s = current state node
            if s is head node of path
                continue
            end
            add (t,s) to pQueue
            if curUTS ∉ UTSHashSet then
                set s abnormal state
                add pQueue to UTSHashSet
            end
        end
    end
    return IPG
end
```

그림 7. 비정상 천이 함수를 호출할 노드를 지정하는 알고리즘.

Fig. 7. Algorithm to set node which abnormal transition will be occurred.

```
ts = genTestSuite(MST,IPG)

input     MST, IPG
output    ts  : test suite
begin
    testSuite = {∅}
    for each path in IPG
        testCase = {∅}      // init test case
        for each node in path
            s = current state node
            tn = next transition
            sn = next state
            if s is abnormal transition state
                for each transition ta in MST
                    add (s, ta, s) to testCase
                end
            end
            if s is end node of path
                break
            end
            add (s, t, sn) to testCase
        end
        add testCase to testSuite
    end
    return testSuite
end
```

그림 8. IPG 기반 테스트 스윗을 생성 알고리즘.

Fig. 8. Test suite generation algorithm based on IPG.

IV. 구현 및 실험

이 절에서는 그림 1의 FSM과 OPRoS 컴포넌트의 FSM에 제안방법과 [8]에서 제안한 방법을 이용하여 테스트 스윗을 생성한 결과를 제시하고 비교한다. 그리고 OPRoS 컴포넌트에 제안한 알고리즘을 적용하여 실제 컴포넌트의 상태 테스트를 수행한 결과를 보인다.

[8]과 [9]에서는 일부 변이 연산자에 의해 생성된 변이 상태도를 원래의 상태도와 차별화시키기 위해 수동으로 테스트 스윗을 작성해야 하는 경우가 있었다. 이 논문에서 제안하는 방법은 테스트 과정의 자동화가 가능하므로 이클립스 플러그인 형태의 테스트 도구를 개발하였고, 이 자동화 도구를 그림 9에 표시하였다.

구현된 자동화 도구에서는 XML 형태로 정형화시킨 FSM을 사용하여 다양한 컴포넌트 모델에 대해 알고리즘을 적용할 수 있도록 하였다. 테스트 도구를 이용하면 테스터는 대상 FSM을 기술한 XML 파일을 선택하고 패러미터를 지정하여 테스트 스윗을 생성할 수 있다. 그리고 컴포넌트의 프로젝트와 테스트 스윗을 선택하여 자동으로 테스트를 수행하고 결과를 확인할 수 있다. 그림 6의 XML 스키마를 사용하여 그림 1의 FSM을 XML로 표현하면 그림 10과 같이 표현할 수 있다. 그림 11에는 OPRoS 컴포넌트의 FSM 모델[1,11]이 제시되어 있다.

실험에서는 그림 1의 예제와 OPRoS 컴포넌트의 FSM 모델[1,11]을 대상으로 왕복 커버리지 기준을 사용해 테스팅 트리를 생성하였다. 왕복 커버리지 기준은 FSM을 순회해 같은 노드를 두 번 만나는 것을 기준으로 하고, 이 기준을 만족하게 되면 순회는 종료된다. 실험에서는 비교적 단순한 FSM모델에 대하여 다양한 테스트 케이스를 생성하기 위해 왕복 커버리지 기준에 왕복수준(RT-level) 패러미터를 도입하였다. 테스팅 트리 생성시 이 패러미터를 적용하여 순회를 종료하기 위한 같은 노드를 만나는 횟수를 두 번 이상으로 조정할 수 있게 하였다. 이 논문에서 제안한 알고리즘과 [8]의 MNR 변이 연산자를 이용하여 테스트 스윗에 비정상 천이를 삽입하여 수행한 결과가 각각 표 2에 제시되어 있다. 전자의 결과는 N_{prop}로, 후자의 결과는 N_{MNR}로

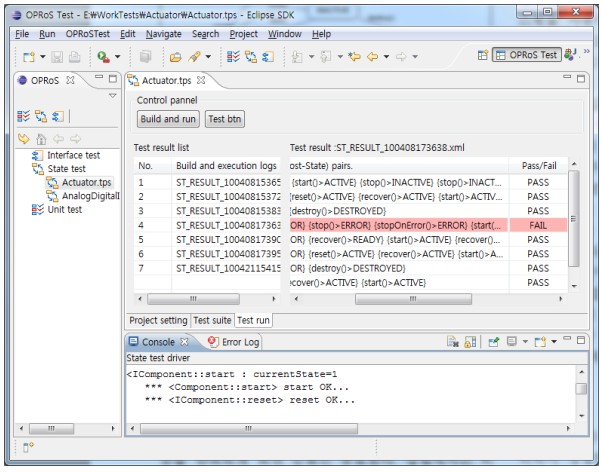


그림 9. 이클립스 기반 상태 테스팅 자동화 도구.

Fig. 9. State testing automation tool based on Eclipse.

```
<FSM>
  <stateList>
    <state name="s1" type="INIT">
      <transitionList>
        <transition trigger="t1" goto="s2" />
      </transitionList>
    </state>
    <state name="s2">
      <transitionList>
        <transition trigger="t2" goto="s4" />
        <transition trigger="t3" goto="s3" />
      </transitionList>
    </state>
    <state name="s3">
      <transitionList>
        <transition trigger="t4" goto="s3" />
        <transition trigger="t5" goto="s2" />
      </transitionList>
    </state>
    <state name="s4">
      <transitionList>
        <transition trigger="t6" goto="s3" />
        <transition trigger="t7" goto="s5" />
      </transitionList>
    </state>
    <state name="s5" type="FINAL" />
  </stateList>
</FSM>
```

그림 10. 그림 1의 FSM 모델의 XML 표현.

Fig. 10. XML represenation of FSM model in Fig. 1.

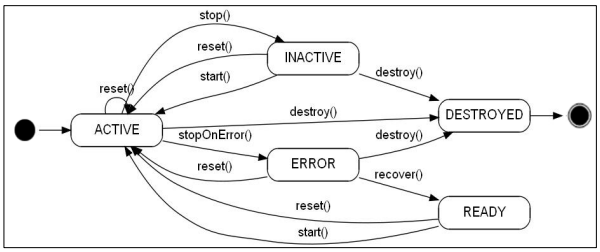


그림 11. OPRoS 컴포넌트의 FSM 모델[2].

Fig. 11. FSM model of OPRoS component[2].

표시되어 있다. 표 2에서 N_{MNR}은 [8]의 MNR 변이 연산자를 FSM 의 모든 상태에 적용하여 생성한 비정상적인 천이의 개수를 의미한다. N_{MNR}은 왕복수준 패러미터에 의해 결정된다. 그리고 UTS의 크기 n은 제안하는 경로 이력 커버리지를 적용해 감소시킨 비정상 천이의 호출 수 N_{prop}를 결정한다.

표 2의 결과를 보면 제안한 경로 이력 커버리지 방법을 사용한 경우 비정상 천이의 개수가 현저히 감소한 것을 확인할 수 있다. 그러나 테스트 스윗을 구성하는 비정상 천이가 줄어든 만큼 테스트 수행시 결함을 검출할 기회를 잃을 수 있다는 것을 직관적으로 알 수 있다. 따라서 합리적인

표 2. MNR 변이 연산자와 경로 이력 커버리지 방법에 대한 비정상 천이 호출 수의 비교.

Table 2. Comparison of abnormal transition call times between MNR mutant operator and path-history coverage.

	왕복 수준	N_{MNR}	UTS-n	N_{prop}	감소율 (%)
그림 1의 예제 FSM	2	**110**	2	**36**	67.3
			3	**46**	58.2
			4	**46**	58.2
	3	**629**	2	**36**	94.3
			3	**66**	89.5
			4	**111**	82.4
OPRoS FSM	2	**59**	2	**16**	72.9
			3	**24**	59.3
			4	**24**	59.3
	3	**681**	2	**16**	97.7
			3	**40**	94.1
			4	**92**	86.5

```
onStart () {
…
if(infraredSensor == NULL ||
    (getStatus() != OPROS_CS_READY
     && getStatus() != OPROS_CS_INACTIVE)) {
        return OPROS_PRECONDITION_NOT_MET;
…
}
```

그림 12. 에러가 발생한 컴포넌트 소스 코드.

Fig. 12. Source code of component which occurs error.

테스트 스윗 생성 방법은 비정상 천이의 수를 줄이면서도 결함 검출 효율은 크게 떨어뜨리지 않아야 한다.

제안한 방법의 유효성을 검증하기 위해 실험에서는 실제 구현된 OPRoS 컴포넌트[12]에 대하여 MNR연산자와 경로 이력 커버리지를 통해 생성한 테스트 스윗을 각각 사용하였다. 테스트 스윗 생성 시 사용되는 입력 패러미터인 왕복 수준과 UTS 크기를 각각 3으로 설정하였다. 이와 같이 생성된 두 개의 테스트 스윗을 총 22개의 OPRoS 컴포넌트에 적용해 테스트를 수행한 결과 4개의 컴포넌트에서 상태천이 에러가 공통적으로 발생하였다. 그리고 각각의 테스트 결과를 분석하여 동일한 결함을 검출할 수 있었다. 제안한 방법을 이용한 실험에서는 테스트 수행시간을 줄일 수 있었고 같은 에러를 중복해서 보고하는 횟수가 줄어들어 결과 분석이 용이하였다. 에러가 보고된 4개의 컴포넌트 모두 그림 12의 컴포넌트 소스코드에서 사용자가 임의로 컴포넌트의 상태천이 조건을 잘못 검사하여 실패 값을 반납하는 코드가 결함으로 분석되었다. 이와 함께 OPRoS 프레임워크의 동작에서도 결함을 발견할 수 있었다. OPRoS의 FSM에서는 READY 상태에서 DESTROYED 상태로 바로 천이할 수 없도록 명시되어 있었으나 READY 상태에서 destroy()함수를 호출할 경우 상태 천이를 일으키는 에러를 확인할 수 있었다.

V. 결론

이 논문에서는 컴포넌트의 상태 천이를 시험하기 위하여 정형화된 FSM을 이용해 테스트 스윗을 생성하는 방법과 테스트 스윗에 비정상적인 천이를 유발시키는 동작을 삽입하고 그 개수를 합리적으로 줄이기 위한 경로 이력 커버리지 방법을 제안하였다. 제안하는 상태 테스트 스윗 생성 방법은 상태 천이를 유발시키는 순서를 생성하기 위한 기존의 커버리지 기준과 결합하여 사용할 수 있으므로 FSM의 특성에 적합한 커버리지를 선택할 수 있다. 또한 테스터는 대상 컴포넌트 FSM 모델의 복잡도를 판단하여 비정상적인 천이의 개수의 감소 정도를 조절함으로서 효율적인 테스트 스윗을 작성할 수 있다.

실험에서는 이 논문의 제안 방법과 [8]의 알고리즘을 그림 1의 예제와 OPRoS의 FSM을 포함한 2개의 예에 적용하여 테스트 스윗을 생성한 결과를 통하여 비정상 천이의 감소정도를 보였다. 동시에 제안한 방법을 적용한 테스트 도구를 구현하여 테스트 스윗을 생성하였다. 이 테스트 스윗을 사용해 실제 OPRoS 컴포넌트에 대한 상태천이 시험을 수행하여 에러를 검출하고 제안한 방법의 유효성을 보였다. 그리고 경로 이력 커버리지를 사용해 동일한 에러에 대한 보고와 테스트 수행시간을 감소시키는 효과를 확인할 수 있었다.

참고문헌

[1] B. Y. Song, S. W. Jung, C. S. Jang, and S. H. Kim, “An introduction to robot component model for OPRoS(Open Platform for Robotic Services),” *Workshop Proceedings of SIMPAR 2008 Intl. Conf.* on Venice, Italy, Nov. 2008.

[2] OPRoS group, www.opros.or.kr

[3] S. H. Park, Y. T. Jin, and S. M. Hwang, “A component testing method using interface information,” *Korea Information Science SocietyS(B),* vol. 30, no. 1, pp. 127-129, 2003.

[4] H. S. Hong, Y. R. Kwon, and S. D Cha, “A state-based testing method for classes,” *Korea Information Science SocietyS(B),* vol. 23, no. 11, pp. 1145-1154, Nov. 1996.

[5] T. S. Chow, “Testing software design modeled by finite-state machines,” *Software Engineering, IEEE Transactions*, vol. SE-4, pp. 178-187, 1978.

[6] J. A. Dallal and P. Sorenson, “Generating class-based test cases for interface classes of object-oriented black box frameworks,” *Proc. of World Academy of Science, Engineering and Technology,* vol. 16, Nov. 2006.

[7] J. Gao, R. Espinoza and J. He, “Testing coverage analysis for software component validation,” *Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th Annual International,* CA, USA, vol. 1, pp. 463-470, July 2005.

[8] M. A. Rho and B. J. Choi, “Class test process applying state-based mutation test criterion,” *Korea Information*

Science SocietyS(B), vol. 25, no. 8, pp. 1206-1217, 1998.
[9] S. C. P. F. Fabbri, J. C. Maldonado, and M. E. Delamaro, "Proteum/FSM: a tool to support finite state machine validation based on mutation testing," *Proc. of the Computer Science Society, XIX International Conference of the Chilean,* pp. 96-104, Nov. 1999.
[10] S. W. Maeng and H. S. Park, "Test suite generation algorithm for component state testing including abnormal transitions," *Proc. of ICROS Annual Conference 2010,* Chuncheon, Korea, pp. 137-138, May 2010.
[11] OPRoS grp, http://203.255.255.39/new/board/?SET=130
[12] KOROS 1067-2:2009, "개방형 로봇 소프트웨어 플랫폼 : 제2부 - 컴포넌트," 한국 로봇협회, 2009.

맹 상 우

2008년 강원대학교 전자통신과 졸업. 2008년~현재 동 대학원 석사과정. 관심분야는 소프트웨어 공학, 소프트웨어 테스팅 자동화

박 홍 성

1983년 서울대학교 제어계측공학과 학사 졸업. 1986년 동 대학원 석사. 1992년 동 대학원 박사. 1992년~현재 강원대학교 전자통신공학과 교수. 관심분야는 로봇 S/W 플랫폼, 무선데이터통신, 실시간 통신.