상호운용성 테스트를 위한 EFSM 기반 행위 모델링

°박형진* 유철중* 장옥배* 홍상기**
*전북대학교 컴퓨터과학과 **한국전자통신연구원
*{ziiny°, cjyoo, okjang}@chonbuk.ac.kr **sghong@etri.re.kr

Behavior Modeling based on EFSM for Interoperability Test

°Hyung-Jin Park* Cheol-Jung Yoo* Ok-Bae Chang* Sang-Gi Hong**
*Dept. of Computer Science, Chonbuk National University
**Electronics and Telecommunications Research Institute

요 약

네트워크의 발달과 함께 현대의 시스템들은 다른 시스템들과 서로 연동되어 운영되는 경우가 증가하고 있다. 그래서 시스템 자체의 테스팅뿐만 아니라 다른 시스템과의 연동에 관한 상호운용성 테스팅 또한 중요하게 인식되고 있다. 기존 상호운용성 테스팅에 관련된 연구들에서는 EFSM(Extended Finite State Machines)을 기반으로 행위를 모델링한 상태전이 정보를 토대로 테스트 케이스를 생성하는 연구가 많이 진행되어 왔다. 그러나 모든 범위의 상호운용성 테스팅에 적용할 수 있는 체계적인 테스트 프로세스에 대한 연구 및 테스트 케이스 생성의 기반이 되는 EFSM 모델 생성에 관한 체계적인 연구 또한 미흡한 실정이다. 본 논문은 완전한 상호운용성 테스팅을 위해 EFSM 기반 행위 모델링 기법을 제안하고 있다.

1. 서 론

네트워크의 발달과 함께 현대의 시스템들은 다른 시스템들과 서로 연동되어 운용되는 경우가 많아졌다. 전자상거래의 활성화로 기업들은 시스템이나 애플리케이션을 다른 기업들의 시스템과 서로 연동하여 운영하는 경우가 많아졌고, 발전하고 있는 무선 네트워크는 기존 유선 네트워크 상의 프로토콜이나 시스템들과 연계되고 있는 추세이다. 이리하여 시스템 자체의 테스팅은 물론이고, 다른 시스템들과의 연동에 관한 상호운용성 테스팅 또한 그 중요성이 증가하고 있다.

그러나 상호운용성 테스팅에 관한 기존 연구들을 살펴보면 특정 도메인에 국한되어 있거나, 자동화되지 못한 발견적 방법인 경우가 많다[1][2][3]. 자동화되지 못했다는 것은 발생 가능한 모든 경우의 완전한 테스팅을 할 수 없다는 것을 의미하고, 이는 내재되어 있는 오류의 가능성을 발견할 수 없게 만든다. 그리고 더 나아가 중복된 테스트 케이스로 인해 테스팅의 효율을 떨어뜨릴 것이다. 위의 문제점들은 두 가지 원칙에 관련된 문제들로 요약될 수 있는데, 그것이 바로 완전성과 중복성 배재의 원칙이다. 완전성이란 모든 상호운용이 테스트되어야 한다는 것이고, 중복성 배재는 전체 테스트의 횟수를 최소화할 수 있도록 모든 중복된 테스트들은 배재되어야한다는 것을 의미한다. 체계적인 상호운용성 테스트를 위한 프로세스는 이 두 가지 원칙이 반드시 지켜져야 한다.

완전성의 원칙을 충족시킬 수 있는 테스트 케이스들을 생성하기 위해서는 먼저 각 시스템들이 가질 수 있는 상태들과 상태들의 변화를 식별하여야 한다. 이는 시스템들의 상태 변화에 따른 각각의 가능한 상호운용의 경우가 하나의 테스트 케이스가 되기 때문이다. 이러한 상태정보를 추출하기 위해서 유스케이스 명세가 사용될 수 있다. 유스케이스 명세는 시스템이 취해야할 행위에 대한 정보를 포함하고 있기 때문이다.

시스템들의 발생 가능한 상태들을 토대로 그 상태들의 전이를 묘사하는 EFSM은 상호운용성 테스팅에 있어서 완전한 테스팅과 중복성을 배재하는 역할을 한다.

본 논문에서는 테스트 케이스 생성을 위해 유스케이스 명세를 기반으로 시스템들의 가질 수 있는 상태들을 추출하고, 그 상태들 간의 전이를 표현하기위한 EFSM 명세 방법에 대해 논의할 것이다. 테스트 케이스 생성에 관한 완전성 및 중복성 배재의 원칙은 이를 위한 기존의 많은 효율적인 알고리즘들이 소개되어 있기 때문에 본 논문에서는 다루지 않을 것이다.

본 논문의 구성은 다음과 같다. 2절에서는 상호운용성 테스트와 관련된 연구에 대해 살펴볼 것이고, 3절에서는 EFSM 기반의 행위 모델링 기법에 대해 설명할 것이다. 4절에서는 EFSM 기반의 행위 모델링 기법의 적용에 대해 살펴볼 것이다. 그리고 마지막 절에서는 지금까지 논의한 사항들에 대해 결론과 향후연구에 대해 논의할 것이다.

본 연구는 한국전자통신연구원의 위탁과제 "개방형 서버의 상호운용성 테스트를 위한 테스트 DB 생성에 관한 연구"의 일부임

2. 상호운용성 테스트

상호운용성 테스팅과 일반 적합성 테스팅과의 근본적인 차이점은 2개 이상의 시스템들 간의 테스팅에 그 목적이 있다는 것이다. 그리고 상호운용성 테스팅은 테스팅으로 기대되는 행위가 각 시스템의 명세로부터 예측되어야 한다[4]. 이러한 특징으로 많은 상호운용성 테스팅들은 명세로부터 EFSM 기반의 테스트 케이스를 생성하는 방법을 사용하고 있다[5].

현재 정형명세를 기반으로 테스트 케이스를 생성하는 기법에 관한 연구는 활발히 진행되어 왔다[6][7][8]. 정형명세는 시스템 행위에 대한 믿을만한 기술이다. 명세란 시스템이 무엇을 해야 할지를 나타내고 있기 때문에 소프트웨어 테스팅에 있어서도 중요한 역할을 담당할 수 있다. 그래서 정형화된 명세로부터 자동화된 테스트케이스 생성기법에 관한 연구는 많이 찾아 볼 수 있다. 하지만 모든 시스템이 정형명세를 기반으로 설계 및 구현되지는 않기 때문에 이 방법은 모든 범위의 테스팅에 공통적으로 적용될 수 있는 기법은 될 수 없다.

이런 이유로 본 논문에서는 정형화 되지 않은 명세로부터 테스트 케이스 생성의 기반이 되는 상태를 추출하고 그 상태들 간의 전이를 명세하는 방법에 대해 기술한다.

3. EFSM 기반 행위 모델링 기법

EFSM 기반 행위 모델링 기법은 크게 두 부분으로 나눌 수 있다. 첫 번째 작업은 명세를 기반으로 각 컴포넌트들이 가질 수 있는 상태들을 추출하는 과정이다. 다음 작업은 그 상태들을 바탕으로 EFSM 명세를 통해 상태들의 전이를 표현하는 과정이다.

그림 1은 위의 EFSM 기반 행위 모델링 기법의 전체적인 흐름을 도식화한 그림으로써 각 단계들과 산출물들 사이의 입출력 관계를 나태내고 있다.

3.1 상태 추출 프로세스

상태 추출 프로세스는 유스케이스 명세에서 상태를 식별하고, 컴포넌트별 속성을 식별하는 작업과 식별된 속성을 이용

하여 상태들을 추출하는 과정으로 구성된다. 그림 2는 그 과정을 보이고 있다.

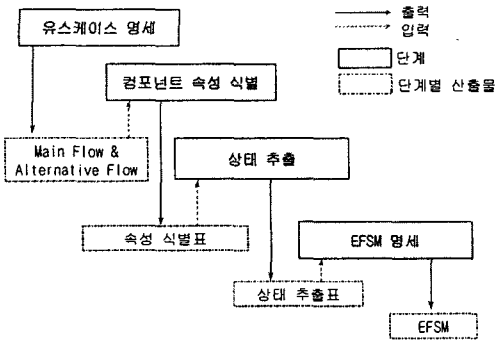

그림 1 EFSM 기반 행위 모델링

Input	**Main Flow & Alternate Flow Statements**
Output	**States**
Step 1	**컴포넌트별 속성 식별표 작성**
	(1) 문장에 포함된 컴포넌트 식별하고 표현된 행위를 각각 해당 항목에 기술 (2) 문장에 조건문이 포함된 경우 해당 조건을 Alternative Thread 항목에 기술 (3) 표현된 행위에 따라 해당 컴포넌트의 속성 식별 및 가능한 속성 값 기술(Alternative Thread에 해당하는 속성의 경우 속성 이름 앞에 'A+*number*' 형식의 키워드 기술) (4) 행위 전후의 컴포넌트 속성 값 변화에 따른 해당 컴포넌트의 상태이름 정의(Alternative Thread에 해당하는 속성 값의 변화는 고려하지 않음) (5) 모든 문장에 대하여 (1)~(4) 단계를 수행한 경우 종료되며, 그렇지 않은 경우 (1) 단계로 되돌아감
Step 2	**속성 식별표를 이용한 상태 추출표 작성**
	(1) Step 1에서 추출된 속성 식별표를 통하여 후보 속성들 및 소속 컴포넌트들을 해당 항목에 기술 (Alternative Thread에 해당하는 속성 값은 상태와 무관하기 때문에 해당 속성으로 고려하지 않음) (2) 유스케이스 문서의 행위에 따른 후보 속성들의 값을 해당 항목에 기술 (3) 속성 값들을 나타낸 각 열을 하나의 상태로 간주하여 이를 시스템의 상태로 식별하고 상태 id를 부여함 (4) 각각의 상태 id별로 상태 이름을 부여함

그림 2 상태 추출 프로세스

상태 추출 프로세스에 의해 3가지 표가 작성되는데 각 표들의 구성은 다음과 같다.

* 컴포넌트 구성표
- *component name*: 문장에서 식별된 컴포넌트 이름
- component description 기술: 컴포넌트에 대한 간단한 설명

* 컴포넌트별 속성 식별표
- statement no.: 유스케이스 명세 문장의 번호
- component: 문장에 포함되어있는 컴포넌트
- behaviors: 컴포넌트가 취하는 행위
- alternative thread: 문장에 포함된 조건
- attributes: 행위의 속성
- component state: 속성값이 변했을 때 컴포넌트의 상태 이름

* 상태 식별표
- component name: 문장에서 식별된 컴포넌트 이름
- attribute: 해당 컴포넌트의 속성
- attribute value: 속성값
- state no.: 각 속성값들의 조합으로 생성된 상태

다음 절에서는 본 절에서 제안한 기법을 적용한 사례를 제시한다.

3.2 EFSM 명세

EFSM(Extended Finite State Machines)은 다음과 같이 구성되고,

$$M=(I, O, S, \vec{x}, T)$$

각 튜플의 의미는 다음과 같다.

I: 입력심볼, O: 출력심볼, S: 상태, $\vec{x}$:변수, T: 전이

집합 T의 각 전이 t는 다음과 같은 6개의 튜플로 이루어지고,

$$t=(s_t, q_t, a_t, o_t, P_t, A_t)$$

각 튜플은 초기상태, 종료상태, 입력, 출력, 술어, 동작을 의미한다. 변수들과 함께 확장된 FSM은 시스템의 행위를 모델링하는데 많이 사용되는 기법이다. 본 연구에서는 변수(variable)가 속성을 나타낸다.

4. EFSM 기반 행위모델링 기법의 적용

3절에서 제안한 EFSM 기반 행위 모델링 기법을 유스케이스 명세로부터 EFSM 명세까지의 과정을 단계별 산출물들을 통해 알아본다. 적용해볼 대상은 위치기반 시스템에서 서비스 제공자가 현재위치를 요청하기 위해 그에 필요한 XML 문서를 작성하고 OMC의 인증을 통해 LGC에게 서비스 처리를 요청하는 과정을 기술한 유스케이스 명세의 일부분이다.

표 1 현재위치 요청 유스케이스 명세

Use case	현재위치 요청
Brief description	위치정보 요청 및 제공에 관한 기록을 관리
Actors	SP
Preconditions	MLP3.0을 준수
Main flow	1. SP는 현재위치 요청을 위한 XML 문서 생성 2. SP는 SOAP 방식으로 플랫폼에 메시지 전송 3. OMC는 메시지를 수신하고 분석 4. OMC는 인증된 서비스인지 검사 5. OMC는 인증 실패시 에러 메시지 반환 6. OMC는 LGC에 TCP/IP를 통해 서비스 처리 요청
Alternative flows	...
Postconditions	...

4.1 컴포넌트 식별 및 컴포넌트별 속성 식별

상태 추출 프로세스의 Step 1에 의해 유스케이스 명세로부터 '컴포넌트 구성표'와 '컴포넌트별 속성 추출표'를 작성할 수 있다. 표 2와 표 3은 이와 같이 작성된 컴포넌트 구성표와 컴포넌트 속성 추출표를 나타낸다. 컴포넌트 구성표는 문장 중에서 행위의 주체가 될 수 있는 컴포넌트들을 식별해놓았다. 컴포넌트별 속성 추출표에는 각 컴포넌트의 행위에 해당하는 행위와 그 행위의 속성들을 식별하고, 속성값의 변화에 따른 컴포넌트 상태의 이름을 정의한다.

표 2 컴포넌트 구성표

Component Name	Component Description
SP	LBS 응용 서비스 제공자
OMC	플랫폼 운영 및 관리 시스템
LGC	위치획득 게이트웨이 시스템

4.2 상태 추출

상태 추출 프로세스 Step2에 의해 컴포넌트별 속성 추출표를 바탕으로 표 4 상태 식별표를 작성할 수 있다. 상태 추출표는 컴포넌트의 속성값이 변함과 함께 그 속성값들의 세로열

조합으로 하나의 상태가 결정되는 것을 나타낸다.

표 4 상태 식별표

Comp. Name.	Attributes	Attribute Values				
SP	Idle_SP	T	F	F	F	F
	CreateXML_SP		T	T	T	T
	SendMsg_SP		F	T	T	T
OMC	Idle_OMC	T	T	F	F	T
	ReceiveMsgFromSP_OMC			T	T	T
	ReqServToLGC_OMC			F	A1=T	A1=F
					T	F
	retErr_OMC				F	T
State No.		S1	S2	S3	S4	S5

4.3 EFSM 명세

상태 추출 프로세스로부터 추출된 상태 S1, S2, S3의 상태전이는 그림 3과 같은 EFSM 명세를 작성할 수 있다. SP가 휴면 상태에서 XML 문서를 생성해서 메시지를 보내는 상태의 전이를 정의한다.

5. 결론 및 향후연구

상호운용되는 시스템들이 증가함으로써 상호운용성 테스트가 중요하게 인식되고 있다. 그러나 기존의 테스팅 방법은 특정 범위에 국한되거나, 특정 시스템에 대한 발견적 방법에 의존하는 경우가 많았다. 그래서 본 논문에서는 모든 범위의 상호운용성 테스팅에 적용될 수 있는 체계적인 테스트 프로세스를 위해 정형화되지 않은 유스케이스 명세로부터 상태들을 추출하고 그 상태들의 전이를 모델링하는 방법에 대해 논의하였다.

본 연구에서 제안한 EFSM은 완전성과 중복성 배재원칙을 따르는 효율적인 테스트 생성을 위한 기초가 되는 데이터이다. 향후 연구에서는 본 연구에서 제안한 EFSM을 토대로 최적화된 테스트 케이스를 추출하기 위한 효율적인 알고리즘에 관한 연구와 이 알고리즘을 기반으로 테스트 케이스를 자동생성하는 기법 및 도구개발에 관한 연구가 필요하다.

참고문헌

[1] J. Gadre, Rohre C, C. Summers, and S. Symington. "A COS study of OSI interoperability," Computer Standards and Interface, vol. 9(3), pp. 217-237, 1990

[2] G. S. Vermeer and H. Blik., "Interoperability testing: Basis for the acceptance of communicating systems," VI(C-19), Elsevier Science Publisher B.V., 1994

[3] Pei Hsia and David Kung, "software requirements and acceptance testing," Annals of Software Engineering, vol. 3, pp. 291-317, 1997

[4] Sungwon Kang, "Relating interoperability testing with conformance testing," Global Telecommunications Conference, vol. 6, pp. 3768-3773, 1998

[5] Griffeth, N., Hao, R., Lee, D., Sinha, R.K., "Interoperability testing of VoIP systems," Global Telecommunications Conference, vol. 3, pp. 1565-1570, 2000

[6] Miao Huaikou, Liu Ling, "A test class framework for generating test cases from Z specifications," Engineering of Complex Computer Systems, pp. 164-171, 2000

[7] Cunning, S.J., Rozenblit, J.W., "Automatic test case generation from requirements specifications for real-time embedded systems," IEEE SMC '99 Conference Proceedings, vol. 5, pp. 784-789, 1999

[8] Cunning, S.J., Rozenbiit, J.W., "Test scenario generation from a structured requirements specification," Engineering of Computer-Based Systems, pp. 166 - 172, 1999

표 3 컴포넌트별 속성 식별표

Stat. No.	Comp.	Behaviors	Alternative Thread	attributes	component state
1	SP	create XML(SP)		idle_SP{T, F} createXML_SP{T, F}	T : idle_SP, F : T : createXML_SP, F : XMLCreatFail_SP
2	SP	send message to OMC(SP)		sendMsg_SP{T, F}	T : sendMsg_SP, F : idle_SP
3	OMC	receive message from SP(OMC)		idle_OMC{T, F} receiveMsgFromSP_OMC{T, F}	T : idle_OMC, F : T : receiveMsgFromSP_OMC, F : idle_OMC
4	OMC	authenticate service(OMC)		A1authServSucc_OMC{T, F}	
5	OMC	return error(OMC)	인증 성공여부	A1authServSucc_OMC{T, F} retErr_OMC{T, F}	T : retErr_OMC(A1), F : requestServToLGC_OMC(A1)
6	OMC LGC	request service to LGC(OMC)		A1authServSucc_OMC{T, F} reqServToLGC_OMC{T, F} idle_LGC{T, F} receiveReqFromOMC_LGC{T, F}	T : reqServToLGC_OMC(A1), F : T : idle_LGC, F : T : receiveReqFromOMC_LGC, F : idle_LGC

```
Initial state: S1(Idle_SPIdle_OMCIdle_LGCIdle_MODBIdle_DB)
Initial value of variables:
        ......
//Format
//From-State
//{Input}/{Output}/{Predicates}/{Actions}/{Color} Next-State
//
transition {
Idle_SPIdle_OMCIdle_LGCIdle_MODBIdle_Carrier
  {clientReq}/{}/{}/{Idle_SP=false; CreateXML_SP=true}/{White} S2(CreateXML_SP)
CreateXML_SP
  {XML}/{SOAPMsg}/{}/{sendMsg_SP=true; Idle_OMC=false; ReceiveMsgFromSP_OMC=true}/{Black} S3(SendMsg_SPReceiveMsgFromSP_OMC)
SendMsg_SPReceiveMsgFromSP_OMC
  {SOAPMsg}/{reqServ}/{A1}/{ReqServToLGC_OMC=true; Idle_LGC=false; ReceiveReqFromOMC_LGC=true}/{Black} S4(ReqServToLGC_OMCReceiveReqFromOMC_LGC
  {SOAPMsg}/{Err}/{!A1}/{retErr_OMC=true; Idle_OMC=true}/{Black} S26(retErr_OMCIdle_OMC)
        ......
  }
```

그림 3 EFSM 명세