情報保護學會論文誌
第19卷 第4號, 2009. 8

암호모듈 검증을 위한 UML 2.0 상태도 기반의 유한상태모델 명세 및 분석*

이 강 수,† 정 재 구, 고 갑 승‡
한남대학교

UML 2.0 Statechart based Modeling and Analysis of Finite State Model for Cryptographic Module Validation*

Gang-soo Lee,† Jae-Goo Jeong, Kab-seung Kou‡
Hannam University

요 약

암호알고리즘 및 암호함수를 하드웨어적 또는 소프트웨어적으로 구현한 암호모듈을 암호모듈검증체계(Cryptographic Module Validation Program, CMVP) 내에서 시험(또는 인증, 검증)을 받기 위해서는 암호모듈에 대한 유한상태모델(Finite State Model, FSM)이 개발되고 제공되어야한다. 그러나 FSM을 체계적으로 모델링하고 분석하는 지침은 개발자와 시험자의 경험이므로 잘 알려져 있지 않다. 본 연구에서는 CMVP내에서 암호모듈의 검증을 위해 요구되는 FSM의 모델링, 분석지침, 천이시험경로 생성알고리즘을 제시하고 모델링도구인 CM-Statecharter를 개발하였다. FSM은 UML 2.0의 상태도를 이용해 모델링한다. 상태도는 FSM의 부족한 점을 보완하고 암호모듈의 FSM을 정형적이고 쉽게 명세할 수 있는 모델이다.

ABSTRACT

A cryptographic module (CM) is an implementation of various cryptographic algorithms and functions by means of hardware or software. When a CM is validated or certified under the CM validation program(CMVP), a finite state model(FSM) of the CM should be developed and provided. However, guides or methods of modeling and analysis of a FSM is not well-known, because the guide is occasionally regarded as a proprietary know-how by developers as well as verifiers of the CM. In this paper, we propose a set of guides on modeling and analysis of a FSM, which is needed for validation of a CM under CMVP, and a transition test path generation algorithm, as well as implement a simple modeling tool (CM-Statecharter). A FSM of a CM is modeled by using the Statechart of UML 2.0. Statechart, overcoming weakness of a FSM, is a formal and easy specification model for finite state modeling of a CM.

Keywords: CMVP, FSM, UML 2.0, CM

I. 서 론

디지털 형태의 각종 정보에 대한 보안성(즉, 무결성, 가용성, 기밀성)을 보장하는 활동은 정보시스템의 필수업무이다. 특히, 정보기술에 바탕을 둔 암호기술은 보안성을 달성하는 가장 일반적인 방법이 되고 있으며, 각종 암호 알고리즘 및 암호모듈(CM: Cryptographic Module)이 국가기관용 및 민간용으로 사용되고 있다. 여기서, 암호모듈 (또는, 암호제품)은 OpenSSL처럼 암호기능을 제공하는 거대한 소프트웨어 라이브러리로부터 특수 암호용 하드웨어 또는 스

접수일(2009년 4월 13일), 게재확정일(2009년 7월 28일)
* 이 논문은 2008년도 한남대학교 교비 학술연구조성비 지원에 의하여 연구되었음.
† 주저자, gslee@eve.hannam.ac.kr
‡ 교신저자, kabseung@se.hannam.ac.kr

마트카드에 이르기까지 종류와 규모가 다양하다.

개발된 암호모듈은 다른 정보를 보호하는 기능을 가지며 제삼자로부터 보안성이 보증되어야하므로, 시험 및 검증 활동이 활발하다. 미국과 카나다의 '암호모듈검증제도'인 CMVP는 FIPS 140-2 표준[1-3] 및 평가지침인 Derived Test Requirements (DTR)[4,5]에 기반을 둔 제도이며, 우리나라와 일본은 이와 유사한 제도를 운영하고 있다[6- 9]. CMVP에 따르면, 암호모듈을 검증받기 위해서는 암호모듈의 동작이 올바르고 일관성 있게 유한상태모델(FSM: Finite State Model)로 모델링해야한다.

그러나 암호모듈을 FSM으로 모델링하고 검증하는 절차와 지침이 부족하므로, 개발자와 시험자의 경험에 의존하고 있다. 특히, 모듈의 FSM은 대외비이며 지적 자산으로 취급하는 경우가 많으며, 모듈의 동작 상태를 올바르고 완전하게 모델링했는지를 검증하는 방법은 평가기관의 노하우이므로, 구체적인 방법을 공개하지 않고 있다. 한편, Unified Modeling Language (UML) 기반의 분석 및 설계모델에서는 FSM을 대신하여 '상태도'(Statechart, State diagram 또는 State machine diagram)를 이용해 표현하고 있으므로, 모듈을 검증할 때 FSM 대신 상태도를 활용하면 좋을 것이다.

이와 같은 배경에서, 본 논문에서는 CMVP하에서 암호모듈을 위한 FSM을 UML 2.0의 상태도로 모델링하고 이를 검증하는 지침과 천이시험경로 생성 알고리즘을 제시하고 상태도 작성 도구인 CM-Statecharter를 개발하였다. 본 연구의 결과는 CMVP하에서 모듈 개발자 및 벤더뿐 만 아니라 시험자가 활용할 수 있을 것이다.

본 논문의 2장에는 CMVP에서 FSM의 모델링 및 평가 요구사항을 분석하고 기존의 상태도 생성 방법들을 조사한다. 3장에서는 암호모듈의 FSM을 모델링할 수 있도록 UML 2.0의 상태도를 정의하고 암호모듈의 상태도를 명세하는 방법을 제시하며, 4장에서는 상태도를 검증하는 방법과 천이시험경로 생성 알고리즘을 제시한다. 5장에서는 상태도를 작성하는 도구인 CM-Statecharter를 설명하며, 6장에서 분석과 결론을 맺는다.

II. 관련연구

2.1 유한상태기계와 FSM

유한상태기계와 FSM은 용도 면에서 다음과 같은 차이가 있다[12,13]. 첫째, 유한상태기계는 유한개의 상태와 천이(transition)로 구성된 가장 간단한 오토마타이다. 현 '상태'에서 (주어진 '행동'을 수행하고[옵션]) 지정된 '사건'이 발생하면 (즉, '가드'(조건)가 만족되면[옵션]), '천이'에 부여된 행동을 수행한 후 다음 상태로 변한다. 여기서, 행동이란 오토마타가 구동기에 제어 신호를 보내는 것이다. 이 경우, 오토마타는 제어기의 역할을 하며 제어용 프로그램에 해당한다. 메모리 없는 하드웨어 장치를 제어할 때 주로 사용한다. 둘째, FSM은 어떤 시스템(예: 암호모듈)의 동적인 변화를 상태와 천이를 통해 모델링한 것이며, 동적인 요구사항을 명세하고 분석할 때 사용한다.

상태도나 패트리넷은 확장된 FSM이며, 오토마타 이론에 의하면 유한상태기계(최하위)와 튜링기계(최상위)의 중간 수준의 모델이다[12-15]. 유한상태기계는 모델링 능력은 낮지만 분석이 용이하며, 튜링기계는 모델링 능력은 높지만 분석이 어렵고, 패트리넷은 그 중간에 해당한다. CMVP에서는 모듈의 상태 변화를 FSM을 통해 표현하고 분석할 것을 요구하고 있다.

2.2 암호모듈 검증과 FSM

2.2.1 CMVP에서 FSM 요구사항 분석

FIPS 140-2는 모듈을 검증할 때, FSM을 포함해 11가지 요구사항 클래스와 4가지 보안등급으로 구성된다[1,6]. FSM은 모든 등급에 공통적으로 적용되므로, 모듈의 검증을 위해서는 필수적이다. CMVP에서는 미국, 카나다, 영국, 독일, 일본, 대만에 위치한 15개소의 암호모듈시험소에 의해 1110종의 암호모듈이 검증되었다. 우리나라는 11종, 일본의 JCMVP에서는 9종이 검증되었다.(2009년 4월 1일 기준)[9-11]

FIPS 140-2의 평가지침인 DTR에 의하면, FSM에는 다음과 같이 12가지 상태 (=모드)를 포함할 것을 권장하고 있다[4,7]: ① 전원켜짐, ② 전원꺼짐, ③ 암호관리자, ④ 키/핵심보안매개변수(CSP)주입, ⑤ 사용자, ⑥ 자가시험, ⑦ 오류, ⑧ 우회 [옵션], ⑨ 유지보수 [옵션], ⑩ 일반 초기화 (140-3에서 추가), ⑪ 검증대상 (140-3에서 추가), ⑫ 휴면 (140-3에서 추가) [옵션]

이들 표준문서에서는 FSM을 위한 기본 요구사항만 나타나 있을 뿐 세부적인 모델링 및 분석지침이 없다.

2.2.2 검증제품의 실제 FSM 분석

검증된 모듈의 주요 내용은 "비재산권적 보안정책 문서"를 통해 공개한다. 이 문서는 모듈의 사용자나 구매자에게 정보를 제공한다. 문서에는 모듈의 요구사항들이 요약되어 있다. 일부 모듈의 보안정책문서에는 FSM이 수록되어 있지만 내용이 불충분하며, 대부분의 CM은 FSM을 대외비로 취급하고 있다. 본 연구에서는 미국 CMVP, 한국 KCMVP, 일본 JCMVP에서 검증된 1110개의 암호모듈의 보안정책문서중 22개의 FSM을 조사하였다.

조사결과, 대부분의 모듈은 상세한 FSM을 공개하고 있지 않으며, 대외비로 취급하고 있다. FIPS 140-2에서 표준화한 상태를 기반으로 하지만, 일치하지는 않는다. 즉, 모듈마다 FSM이 매우 다양하다. 상태도를 이용하지 않고 있으며, 모델링 지침이 없고 병행상태와 합성상태가 없다. 작성이 어렵고 가시성이 부족하며 FSM의 검증방법이 나타나 있지 않다.

2.3 FSM과 상태도 모델링방법

2.3.1 직접 FSM 생성 방법

암호모듈의 동작을 FSM이나 상태도로 모델링하는 구체적인 방법이나 지침은 매우 부족하므로, 본 논문에서는 이 문제를 주로 다룬다.

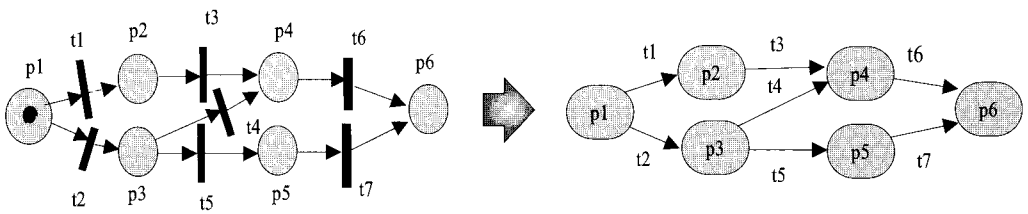
2.3.2 변환에 의한 FSM의 생성방법

시스템의 개발 생명주기 동안 요구사항 명세/분석 및 설계를 위해 작성했던 페트리넷, 유스케이스 다이어그램 또는 순차도 등을 상태도로 변환한다. 이 접근방법은 가장 활발히 연구 중이며[24-28], UML을 이용하여 시스템을 개발할 때에 적합하며, 모델링된 상태도의 정당성은 변환에 사용된 모델(예: 유스케이스 다이어그램 또는 순차도)의 정당성에 영향을 받는다. 그러나 UML을 이용하여 개발하지 않는 대부분의 모듈은 이 접근방법을 사용하기 어렵다.

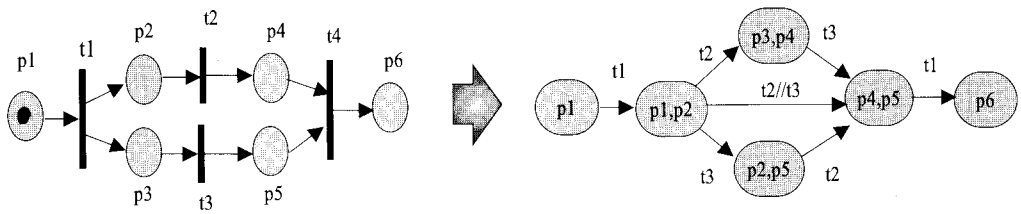
페트리넷을 FSM으로 변환하는 연구의 경우, 페트리넷의 초기마킹을 이용하여 수행(즉, 연속적인 점화) 후 얻은 도달성그래프는 곧 FSM이 된다[25]. 페트리넷이 free-choice net(즉, fork/join 천이가 없는 것) 경우, 수행을 하지 않고서도 도달성그래프를 얻을 수 있다.([그림 1(a)]). 그러나 일반 페트리넷의 경우, 수행을 통해서 도달성그래프(즉, FSM)를 구할 수 있다([그림 1(b)]).

2.3.3 CMVP하에서 FSM 생성법의 문제점

구체적인 FSM 모델링 지침(예: 상태와 천이 간의 관계 등)이 부족하며, 초기상태, 종료상태, 합성상태, 동시상태 등을 모델링하지 않으므로, 모델링 능력이 부족하다. 또한, 상태의 표현에 일관성이 부족하고 상태의 수준과 관점이 다르다. 상태와 사건간의 구분이



(a) free-choice net 수준의 패트리넷 (여기서, 패트리넷 = 도달성그래프 = FSM)



(b) 일반 패트리넷

[그림 1] 패트리넷을 FSM으로 변환

〔표 1〕 패트리넷, FSM, 상태도 및 자료흐름도간의 대응성

모델 / 개념	패트리넷	FSM	상태도	프로우챠트	자료흐름도
상태	플레이스	노드(=상태)	노드(=상태) (합성상태 있음)	-	프로세스
천이	기능을 모델링하며, 화살표는 토큰의 이동을 모델링	화살표는 활동의 수행과 천이를 모델링	화살표는 활동의 수행과 천이를 모델링	화살표는 제어의 흐름을 모델링	화살표는 자료와 자료의 이동을 모델링
활성 상태	토큰이 포함된(마킹된) 플레이스	활성노드(상태)	활성노드(상태)	수행중인 노드	프로세스의 수행 상태
전체 상태	마킹셋	구성(config.)	구성	없음	없음
병행성 모델링	병행 모델링 가능	없음	병행 모델링	병행모델가능	없음
초기상태	초기마킹 (주어짐)	없음	초기상태 표시	시작노드	소스 노드 표시
종료상태	종료마킹	없음	최종상태 표시	종료노드	싱크 노드 표시

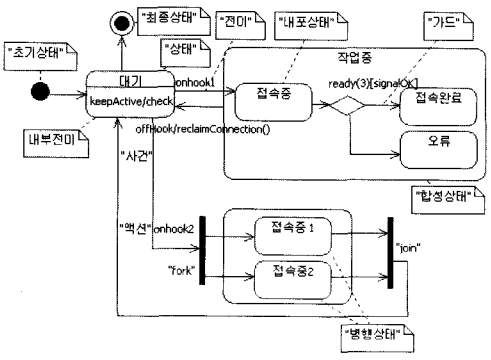
모호하며 중복 상태가 존재한다. 특히, FSM이 '강결합'(strongly component)이 아닌 것이 있으므로, 도달 불가능한 경우가 존재하고 상태도를 이용한 방법이 없다. FSM을 분석하는 방법이 부족하다. 따라서 모듈에 대한 FSM을 모델링하고 분석하기가 어렵다.

상태도의 측면에서, 모델링 방법들은 일반적인(추상적) 사항만을 나타내므로, 암호모듈에 직접 적용하기는 어렵고, 일반적으로 유스케이스 다이어그램과 순차도로부터 변환하여 생성한다. 즉, 이 방법은 UML을 이용한 객체지향 또는 컴포넌트지향의 개발환경에서 적용이 가능하다.

III. 상태도의 정의 및 특성

3.1 정의

일반적인 상태도의 정형적인 정의는 문헌 〔15,23, 26-28〕을 참조한다. 상태도의 기본구조와 요소는 문헌 〔15,25〕의 정의를 따른다. 상태도를 이용해 암호모듈을 개발할 때, UML의 다른 모델과 같이 사용할 수 있다. 〔그림 2〕는 상태도의 예를 보인다.



〔그림 2〕 상태도의 용어 (CM-Statecharter로 작성함)

3.2 표현 및 모델링

대부분의 모듈의 FSM은 '상태천이표'로 표현되며, '현상태', '입력(사건)', '출력(활동)', '다음상태' 열로 구성된다〔1, 4〕. 표의 각 행(튜플)은 "어떤 '현상태'에서 '입력(사건)'이 발생하면, '출력(활동)'을 실행한 후 '현상태'는 '다음상태'로 천이된다."는 정보를 포함한다.

상태도에 대한 상태천이표는 〔표 2〕와 같이 상태도가 표현할 수 있는 정보를 모두 표현할 수 있도록 하였다. 옵션부분을 제외한 부분만을 표현하면, 상태천이표는 FSM에서의 상태천이표와 같아진다.

3.3 모델링 지침

임의의 모듈에 대한 상태도를 모델링하는 일은 정형적으로 나타내기는 어려우므로, 본 연구에서는 '알고리즘'이나 '방법'이라는 용어보다는 '지침' 용어를 사용한다.

3.3.1 기본지침

다수 '객체'(또는, 서브시스템)로 구성된 암호모듈의 경우, 객체마다 별도의 상태도를 작성한다. 모듈의 '상태'는 모듈내의 상태변수('속성')들의 현재 값을 나타낸다. '천이'는 모듈 내 '메소드'의 호출(메시지 도착) 및 수행을 의미하며, 수행 후 '상태'(상태변수의 값)가 변한다. 상태 간 '중복성'이 없고 '누락된' 상태가

〔표 2〕 상태천이표 ('*' 표시는 옵션부분을 나타냄)

현상태 (source target)							천이 Event〔Guard〕/activity 또는 Trigger〔Guard〕/Effect			다음상태 (target state)
	state action									
현상태	*진입동작: 상태 진입시 동작 (한 상태에 진입하는 천이의 영향이 공통일 때 사용) entry/ activity	*진출동작: 상태를 나갈 때 동작 exit/ activity	*내부천이 현상태 내에서 해당사건을 처리 (상태변화 없음) e(a: T) 〔guard〕/ activity	*활동 현 상태에서 수행할 동작 do/ activity	*지연사건 현 상태를 빠져나갈 때 발생한것 처럼 그 효과를 지연시키는 사건	*외부상태의 변경 또는 자체천이를 야기하는 사건에 대한 응답 및 영향(effect). 진입 또는 진출 활동을 야기함 e(a: T) 〔guard〕/ activity	사건 (입력)	*조건 (가드) 이벤트 발생시 '참'이 되어야 함	행동 (출력)	다음 상태
s1	entry/act1	exit/act1		do/ activity1		e(a: T) 〔guard〕/act1	e11	g11	act11	s2
s1	entry/act1	exit/act1				e(a: T) 〔guard〕/act1	e12	g12	act12	s3
..										

〔표 3〕 표준상태 기반 상태천이표 템플리트

현상태 (*표시는 표준 상태)			사건 (입력)	행동 (출력)	다음상태
합성상태	세부상태				
전원	전원켜짐*		"사건을 기입"	"행동을 기입"	"다음상태를 기입"
	전원꺼짐*				
암호관리자 *	암호 초기화*				
	키관리	예비활성화			
		활성화			
		비활성화된			
		파괴된			
		손상된			
		파괴된손상된			
사용자 *	암호연산 수행				
	검증대상 보호함수 수행*				
	비검증대상 보호함수 수행				
주입*	키 주입				
	CSP 주입				
자가시험*					
오류*	자가시험				
	암호키나 CSP 없이 암호화 시도				
	하드 오류				
	소프트 오류				
우회* (옵션)					
유지보수* (옵션)					
일반 초기화*					
휴면*	저전력				
	보류				
	동면				

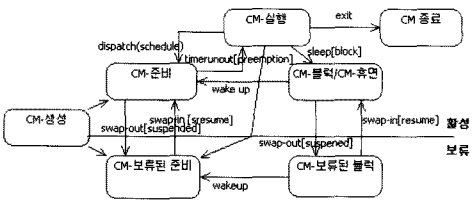
없어야 한다. 상태도는 강결합 그래프이어야 한다. 즉, 모든 각 상태 간 도달 가능해야하며, 이를 위해, 필요시에 천이를 추가해야할 경우가 있다.

3.3.2 표준 제출물로부터 표준 상태를 모델링 (표준상태 기반방법):

FIPS 140과 DTR에서는 19종의 표준 제출물(즉, 암호모듈 검증을 위해 제출해야하는 내용)과 12개의 표준 상태를 제시하였다[1,4]. 상태도 작성자는 표준 제출물로부터 [표 3]에서 보인 '표준상태기반 상태천이표 템플리트'를 사용하여 모델링한다.

3.3.3 프로세스 '생명주기' 상태를 고려 (생명주기 기반 방법)

운영체제처럼 모듈을 생명주기 프로세스 관점에서 보고, 모듈의 상태와 천이를 모델링한다. 이 방법을 이용하면 모듈의 상태도를 쉽고 간단히 작성할 수 있다. [그림 3]은 예를 보인다.



[그림 3] 프로세스 생명주기 기반의 모델링의 예

3.3.4 상태간 관계를 고려

상태간의 관계에 따른 예와 모델링 지침은 [표 4]에서 보인다.

3.3.5 상태의 수준을 통일

상태들의 입자수준을 일정하게 유지한다. 예를 들어, '오류'를 합성상태라 하면 '자가시험오류'는 원자상태이다. 이 경우, '오류'상태는 '시험오류' 및 '운영오류' 등을 포함하므로, '오류'상태는 합성상태로 처리한다.

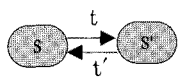
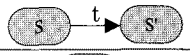
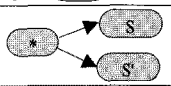
3.3.6 상태와 사건(천이, 조건)간 일관성을 유지

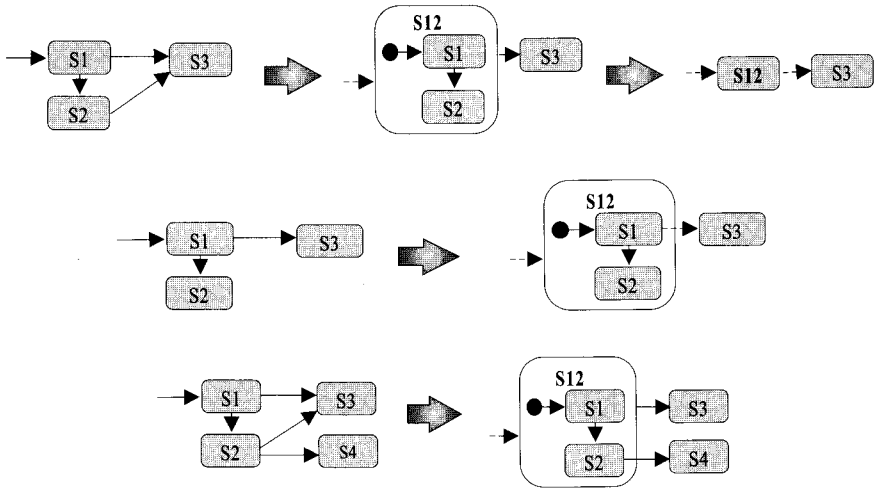
상태와 사건은 관점에 따라 용어가 혼용된다. 예컨대, '경보조건발견'은 상태이기도하고 사건이기도 하므로, 사건을 나타낼 때는 '경보조건발견'으로 하고, 상태를 나타낼 때는 '경보조건처리'로 나타낸다. 또한, '종이걸림'은 사건이기도하고 상태이기도 하므로, 사건을 나타낼 때는 '종이걸림'이며, 상태를 나타낼 때는 '종이걸림처리'이다. 일반적으로, 상태는 지속성(~ing, ~ed)이 있는 상황이며, 사건(천이조건)은 일시성 및 논리성(조건) 을 가진 상황이다.

3.4 상태도 간략화

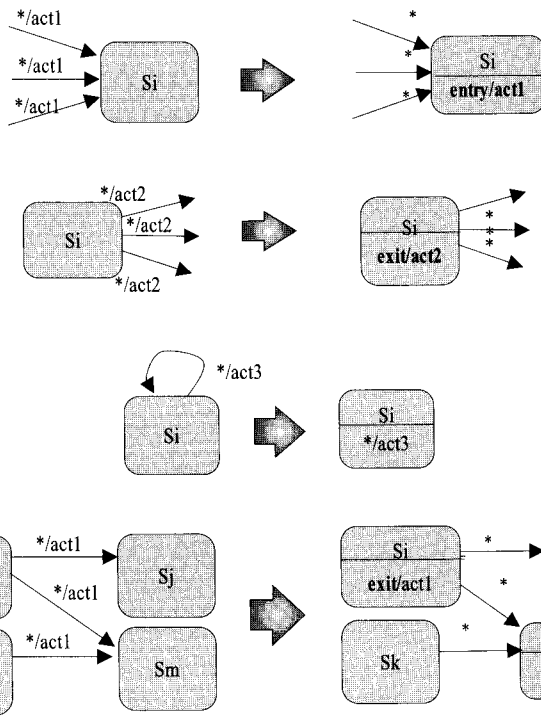
하나의 상태도에 상태 개수가 많으면 이해하고 분석하기가 어려우므로, 공통의 속성을 갖는 2개 이상의 단순상태를 하나의 합성상태로 모델링한다.

[표 4] 상태간의 관계의 예와 모델링 지침

상태간 관계	예	모델링
반사 상태 (SR)	'동작' 상태에서 사건처리후 다시 '동작' 상태로 천이	
대칭 또는 토글상태 (S ⇔ S′)	'전원켜짐'/'전원꺼짐', '검증대상'/'비검증대상', 등. S′=T(S, t), S=T(S′, t′), S′=T(T(S′, t′), t)	
귀결 상태 (원인과 결과) (S ⇒ S′)	'시험'은 원인이며 '시험오류'는 결과임. 순차관계와 같음. S′=T(S, t)	
병행상태 (S ‖ S′)	동시에 시작되고 동기적인 두 상태는 fork/ join천이로 모델링	
선택관계 (S ⊕ S′)	한 상태만 선택되어 활성화	
중복상태 (S ⨟ S′):	S∩S′≠∅이면, S∩S′=∅가 되도록 S′를 수정함	-



〔그림 4〕 합성상태를 이용한 간략화 지침



〔그림 5〕 천이의 간략화 지침

3.4.1 합성상태를 이용한 간략화

대칭상태를 하나로 묶는다(예: 채널1+채널2=채널, on+off=전원, 묵음+소리=소리), 〔그림 4〕와 같이 관련 상태를 하나의 합성상태로 묶는다.

3.4.2 천이의 간략화

〔그림 5〕와같이 상태 내에 entry와 exit를 이용하면 천이를 간략화 할 수 있다.

IV. 상태도 검증방법

상태도(또는, FSM)의 검증과정은 별도의 단계가 아니라, 모델링 단계와 동시에 반복적으로 실시한다. FIPS 140-2의 DTR에는 검증 기준을 추상적으로 제시하고 있으며[4], 본 연구에서는 이 기준과 관련 연구결과[29-31]을 확장하여 암호모듈을 위한 상태도의 검증방법을 체크리스트 형태로 제시한다.

4.1 속성 검증 체크리스트

4.1.1 정확성 및 완전성

모듈의 모든 '상태'(동작 및 오류), '천이', '사건'(입력 및 출력)을 상태도/상태천이표로 모델링했는가? 천이시의 '행동'은 모든 가능한 자료와 제어입력의 결과로써 정의했는가? 모든 '사건'의 입력 파라미터를 체크했는가? 모든 '상태'는 명확히 정의했는가? 한 '사건'은 오직 한 천이만을 트리거 하는가? 1개만의 '초기상태' 및 '최종상태'를 갖는가? 합성 상태는 적어도 1개의 '탈출(exit)상태'를 갖는가? 실세계의 실제 상태는 한 상태로만 모델링했는가? 모든 상태와 천이는 서로 다른 이름을 가지는가? 예외 사례를 적절히 나타냈는가? 모델은 요구사항에 대해 올바르고 완전히 역추적 가능한가?(즉, 요구사항 추적가능성(근원, 이유, 연관성, 구현내용기록)) 암호모듈이 객체지향적으로 개발된 경우, 모듈내의 '메서드'와 FSM내의 '사건'은 대응되는가?

4.1.2 도달성

'초기 상태'(예: 전원켜짐)에서 일련의 정당한 '천이'를 통해 모든 상태에 도달가능한가? 상태도는 강결합 그래프(즉, 모든 노드쌍이 도달가능한 그래프)인가? 부당한 종료 및 핵심적인(문제있는) 상태에 도달하지 않는가? 각 상태에 진입하도록, 모듈을 시험했을 때, 각 상태는 동시에 1개의 상태 표시자를 출력하는가?(즉, 모듈은 한순간에 1개만의 상태에 있음을 나타내는가)

4.1.3 일관성

상태도/상태천이표는 개발자 문서(표준 19종)와 일관성을 유지하는가? 상태도와 모듈의 실제운영과 간에 일관성을 유지하는가? 한 상태 및 사건으로부터 트리거된 2개 이상의 천이의 가드(조건)는 동시에 '참'이 되지 않는가?(즉, 상호 배타적이어야함)

4.1.4 상태간 중복성 및 충돌성

상태들 간에는 중복되지 않는가? 천이간 비결정적 충돌이 없는가? 외부 천이는 내부 천이와 충돌하는가? 외부 천이는 완료 천이와 충돌하는가? 두 완료 천이가 충돌되면, 그들은 동일한 입력 상태를 가지는가?

4.1.5 기타

동기적 상태가 있는 경우, '데드록'이 발생하는가? 반복 가능한 행동은 천이의 사이클 내에서 실행되는가? 오류상태에서 정상(동작 또는 초기화) 상태로의 천이 경로가 여러 개가 있는가? 오류상태에 있을 때를 표시할 수 있는가? 역 행동이 가능한가? 초기상태는 안전한가? 모든 상태에서 '취소' 행동이 가능해야하며, 경고행동은 각 중요행동 이전에 실시하는가? 외부 이벤트는 외부 이벤트보다 높은 우선순위를 갖는가? 모듈에 유지보수 인터페이스 포함 시, 유지보수 상태를 정의했는가?

4.2 천이시험경로 생성

4.2.1 CYC값과 천이시험경로 개수

모듈의 복잡도를 측정하고 시험경로의 개수를 파악하기위해, 상태도에 대한 Cyclomatic Number(CYC)를 이용한다. CYC는 상태도(또는 FSM)내의 '폐쇄 공간수'에 해당하며 '천이개수-상태개수+2'로 계산된다.

McCabe는 CYC값을 통해 플로우챠트의 복잡도와 서로 다른 시험경로 개수를 구하는데 사용하였다[31]. 상태도가 강결합(시작 및 종료노드가 1개씩 있으며, 모든 노드는 도달가능)이면, CYC값은 서로 다른 시험경로의 최대 개수와 같다.

3장의 상태도 작성지침에 따라 모델링된 상태도는 강결합 그래프이므로, 각 상태는 '시작상태'로부터 도달가능하며 천이를 1회씩 시험(즉, '천이 커버리지'가 100%)하기 위한 천이시험경로(Transition Test Path: TTP)의 개수는 상태도의 CYC 보다 같거나 작다. 강결합 그래프가 아닌 상태도의 천이시험 경로수는 CYC값과는 무관하다.

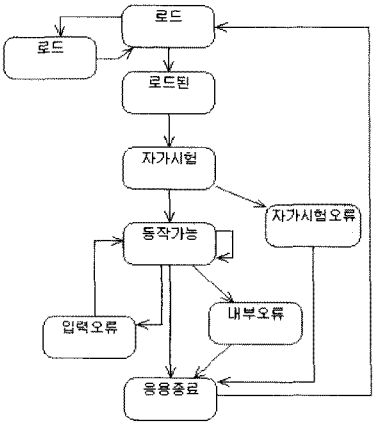
4.2.2 천이시험경로(TTP) 작성법

TTP는 상태도의 모든 천이를 적어도 한번 시험하기위한 경로이며, 상태도의 강결합 여부에 상관없이 다음 알고리즘을 통해 구한다. 알고리즘에서는 "대칭상태" (즉, 두 상태 간에 상호 천이가 있음)와 "반사 천이"(즉, 상태 자신으로 가는 천이)를 별도의 TTP로 간주한다. 본 알고리즘으로부터 생성된 TTP는 생성할 때마다 다른 결과가 나타나므로, 비결정적 알고리즘이다. 모듈의 검증자는 각 TTP에 따라 시험사례를 입력하고 상태도대로 상태가 천이되는지 확인한다. [예 1]과 [예 2]는 천이시험경로 알고리즘의 실행결과를 보인다.

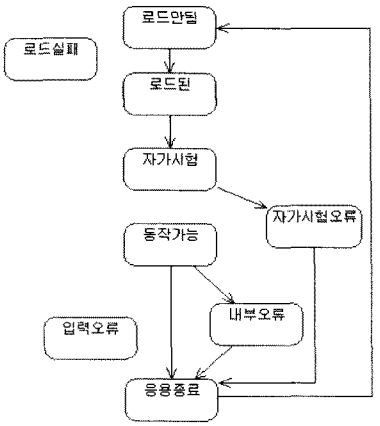
```
[천이시험경로(TTP) 생성 알고리즘]

while (상태도 내에 '대칭상태' 또는 '반사상태' 존재) do
 case '대칭상태': S 및 S' = 대칭 상태명;
     T1 = S에서 S'로 가는 천이;
     T2 = S'에서 S로 가는 천이;
     print S-T1-S'-T2-S;  // T1, T2명칭 없을 때 '*' 출력
     상태도에서 T1과 T2 삭제;
     count = count+1.
  case '반사상태': S = 반사상태;
     print S-T-S; // T는 S의 반사 천이
     상태도에서 T 삭제;
     count = count+1. end
while (상태도 내에 천이가 존재) do
  while (S의 천이 or 사이클 존재) do
     S = 천이가 있는 상태;
     S' = S의 천이에 연결된 상태;
     print S-T-S'; // T는 S에서 S'로의 천이
     상태도에서 T 삭제;
     S = S';                end
     출력에서 연속 중복을 제거;
     print newline;
     count = count+1;   end
CYC = 천이개수 - 상태개수 + 2 // 상태도의 Cyclomatic
number
```

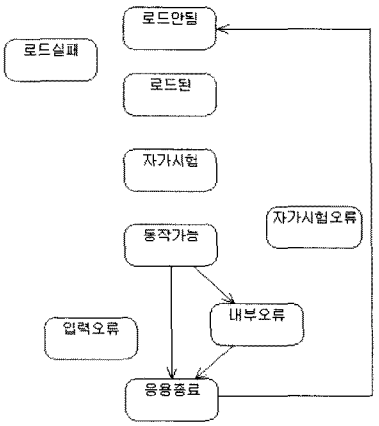
[예 1] [그림 6]은 도시바 솔루션 암호 라이브러리의 상태도에 대한 TTP의 생성과정을 보인다[32]. 이 상태도의 CYC = 14-9+2 = 7이며, 알고리즘으로부터 생성되는 TTP 집합중의 하나는 다음과 같다(※주의, 다수의 경로집합이 존재함). ① 로드안된-1-로드실패-2-로드안된 (대칭), ② 동작가능-7-동작가능 (반사), ③ 동작가능-9-입력오류-10-동작가능 (대칭), ④ 자가시험-5-동작가능-6-자가시험 (대칭), ⑤ 로드안된-3-로드된-4-자가시험-8-자가시험오류-13-응용종료, ⑥ 동작가능-11-응용종료, ⑦ 동작가능-12-내부오류-14-응용종료



(a) 상태도



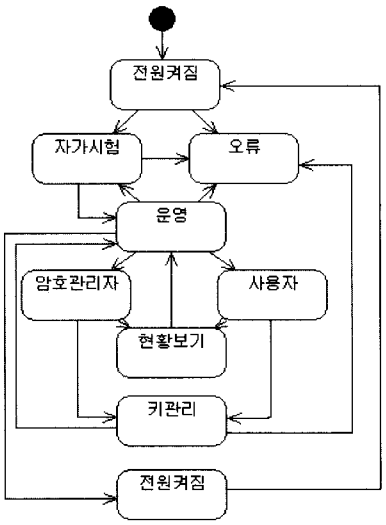
(b)대칭 및 반사경로 출력후



(c) "로드안된-3-로드된-4-자가시험-8-자가시험오류-13- 응용종료" 출력후

[그림 6] TTP 생성과정의 예

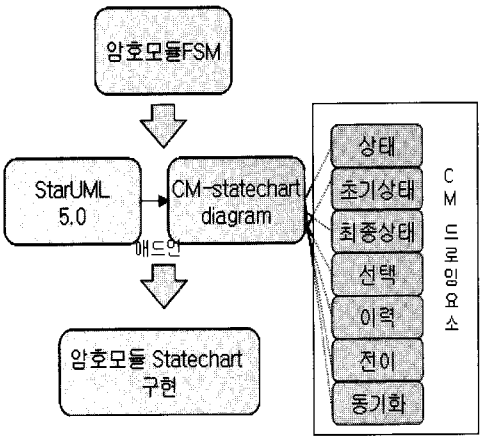
[예 2] [그림 7]은 공개소스 암호모듈인 OpenSSL의 상태도(천이 이름 생략)를 보인다[34]. CYC값은 9 (16-9+2)이며, 다수의 TTP 집합이 존재하며, 그들 중 두 가지는 다음과 같다. ① 자가시험-*-운영-*-자가시험 (대칭), ② 전원켜짐-*-자가시험-*-오류-*-전원켜짐, ③ 운영-*-오류, ④ 운영-*-사용자-*-키관리-*-오류, ⑤ 사용자-*-현황보기-*-운영-*-암호관리자-*-현황보기, ⑥ 암호관리자-*-키관리-*-운영-*-전원꺼짐-*-전원켜짐-*-전원켜짐 및 ① 전원켜짐-*-자가시험-*-오류-*-전원켜짐, ② 운영-*-자가시험-*-운영, ③. 운영-*-암호관리자-*-현황보기-*-운영, ④ 운영-*-전원꺼짐, ⑤ 암호관리자-*-키관리-*-운영-*-오류, ⑥ 운영-*-사용자-*-현황보기, ⑦ 사용자-*-키관리-*-오류



[그림 7] OpenSSL의 상태도

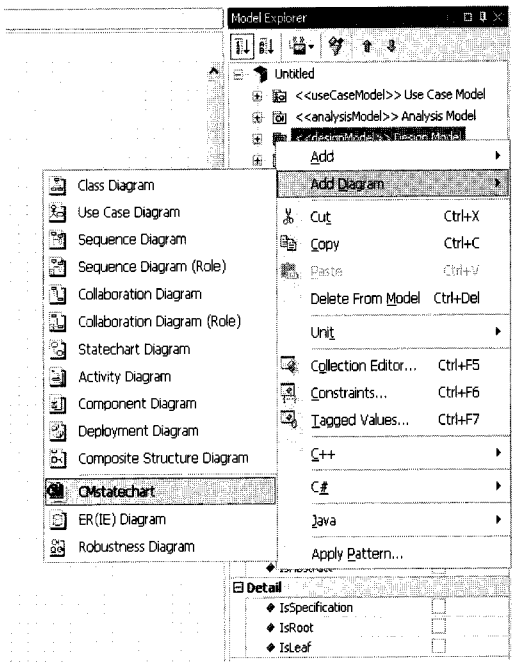
V. 상태도 작성도구 CM-statecharter의 구현

암호모듈 FSM을 위한 상태도 작성 도구인 '암호모듈 상태도작성기(CM-Statecharter)'를 구현하였다. CM-statecharter는 오픈소스 UML 작성기인 StarUML을 기반으로 하며, 모듈의 상태도를 작성하기 위한 애드인 형태의 UML 프로파일로 존재한다[34]. [그림 8]은 CM-statecharter의 구조도이다. 시스템 플랫폼은 Intel pentium 계열의 Windows XP이며, 기반 소프트웨어는 기본적으로 StarUML 5.0이 설치되어야 하며, 기반 스크립트는 XML/XMI이다.

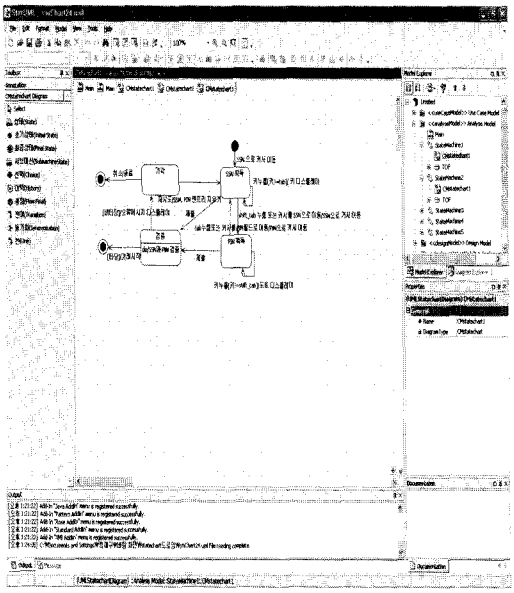


[그림 8] CM-Statecharter 구조도

CM-statecharter를 이용해 암호모듈의 상태도를 모델링하려면 [그림 9]와 같이 우선 StarUML의 Model 메뉴의 Profile에서 CM-statecharter Profile을 불러와야 한다. 이후 ModelExplorer 메뉴에서 CMstatechart를 불러와야 하며, 원하는 드로잉 요소를 선택하여 상태도를 모델링 할 수 있다. [그림 9]는 새로운 CM-statechart를 불러오는 방법이며 [그림 10]은 CM-statechart를 활용한 예시이다.



[그림 9] CM-statechart diagram 추가



〔그림 10〕 CM-statechart 활용 작성 예시

VI. 분석 및 결론

FIPS 140에서 제시한 FSM과 상태도간의 차이점은 〔표 5〕에서 보인다. 상태도는 FSM과 같거나 높은 수준의 정형성을 가지며 모델링이 우수하다. 기존의 UML 모델링 지원도구를 사용할 수 있다. 암호모듈의 동적 모델링 뿐 아니라, 분석 및 설계에도 활용할 수 있다.

〔표 5〕 FSM과 상태도의 비교

모델 / 비교기준	FSM	상태도
상태 (노드)	합성상태 없음	합성상태 모델가능
상태간 관계	XOR	AND, XOR
모델링 능력	작음	높음
모델링 지침	모델링 지침 부족	모델링 지침 있음 (본 연구)
도구	모델링 및 분석도구 부족	UML 도구 다수
가시성	부족	높음
정형성	우수	우수
표준준수	우수	우수
용도	동작모델링용	요구사항 및 설계명세용
개발과 연계성	연계성 부족	중간결과물로 활용

- 제시한 방법과 도구를 이용하면 암호모듈의 FSM을 상태도로 모델링할 때 기존의 문서편집기에 있는 도형편집기를 이용하는 것보다 모델링이 용이하며, 상태도뿐 아니라 유스케이스 다이어그램 및 순차도 등 다른 UML과 함께 사용할 수 있다. 또한, 분석기능도 추가할 수 있다.
- 기존의 FSM보다 모듈의 초기 상태와 최종 상태 또는 합성 상태 등 각종 상태를 구분하는 것이 용이하였다.
- 본 방법은 KS X ISO/IEC 19790의 검증 요구사항을 통해 명시된 대로 FSM 상태표와 상태도 작성을 적용하면서도 확장성이 높은 UML 2.0의 상태차트 모델을 활용하므로, 다양하고 확장성 있는 상태도 모델링이 가능하다(기존 방식은 단순 전이로 표현됨). 따라서 이후 추가적으로 개발되는 암호 FSM에 대한 요구사항을 반영하는 데 유연하게 작용 할 수 있다.
- 개발자와 시험자들이 본 모델 활용 시평가 신청과 시험 평가 수행이 용이 할 것으로 예상된다. 현재 암호 모듈 시험 기준에는 FSM이 암호모듈마다 특성이 다르기 때문에 모듈 시험 시 이에 대한 요구사항이 추상적이고 부분적으로 나타났기 때문에 추가 요구사항을 명시해야 했다. 따라서 표준화 된 본 모델을 활용하면 이러한 문제점이 다소 해결될 것으로 예상된다.

본 연구에서는 CMVP에서 필수적인 업무인 암호모듈의 FSM을 상태도로 모델링하는 지침과 지원도구를 개발하였다. FSM의 단점을 보완하기위해 UML 2.0의 상태도를 이용하였고 암호모듈을 일관성 있고 도달가능한 상태도로 모델링하고 분석하는 지침을 제시하였다. 또한, 상태도를 모델링할 수 있는 도구를 구현하였다. 상태도는 상태와 천이에 여러 가지 조건이나 행동을 모델링할 수 있고, 병행성 있는 상태 및 합성상태를 모델링할 수 있으므로, 기존의 FSM보다 모델링 능력이 우수하며 기존의 UML 모델들로부터 유도할 수도 있다.

본 연구에서도 기존의 실제 FSM을 상태도로 모델링 및 분석하였으나 좀 더 많은 암호모듈에 대해서 모델을 적용해보고 연구에서 제시된 지침과 도구를 개선할 필요가 있다. CMVP에서도 상태도를 FSM을 위한 권고 모델로 지정되도록 해야 한다. 또한, 연구 중 기존의 FSM에서는 많은 문제점을 발견하였으며 이를 해결하기위한 연구가 계속되어야한다.

참 고 문 헌

[1] FIPS PUB 140-2, "Security Requirements for Cryptographic Modules," May 2001.

[2] ISO/IEC 19790: 2006(E), "Security Requirements for Cryptographic Modules," Mar. 2006.

[3] FIPS PUB 140-3(DRAFT), "Security Requirements for Cryptographic Modules," July 2007.

[4] Derived Test Requirements(DTR) for FIPS PUB 140-2, "Security Requirements for Cryptographic Modules," Mar. 2004.

[5] ISO/IEC 24759:2008, "Test Requirements for Cryptographic Modules," July 2008.

[6] 기술표준원, "암호검증기준," KS X ISO/IEC 19790: 2007, 2007년 12월.

[7] 기술표준원, "암호시험기준," KS X ISO/IEC 24759, 2007년 12월.

[8] IT보안인증사무국, "암호검증제출물 작성법 소개," 2007년 2월.

[9] Cryptographic Module Validation Program, http://csrc.nist.gov/groups/STM/cmvp/index.html

[10] 국가정보원 IT 보안 인증 사무국, http://www.kecs.go.kr

[11] Japan Cryptographic Module Validation Program, http://www.ipa.go.jp/security/english/jcmvp.html

[12] D. Drusinsky, Modeling and verification using UML Statecharts, Newnes Pub., Apr. 2006.

[13] A. Jantsch, Modeling Embedded Systems and SOC's-concurrency and time in models of computation, Morgan Kaufmann Pub., June 2003.

[14] D. Harel, "Statecharts: A visual formalism for complex systems," Science of Computer Programming, vol. 8, no. 3, pp. 231-274, July 1986.

[15] J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modeling Language Reference Manual, 2th Ed., Addison-Wesley, July 2004.

[16] I. Kruger, R. Grosu, P. Scholz, and M. Broy, "From MSCS to Statecharts," IFIP WG10.3/WG10.5 International Workshop on Distributed and Parallel Embedded Systems (DIPES'98), pp. 61-71, Oct. 1999.

[17] J. Whittle and J. Schumann, "Generating statechart designs from scenarios," 22'nd ICSE, pp. 314-323, Apr. 2000.

[18] N. Mansurov and D. Zhukov, "Automatic synthesis of SDL models in use case methodology," 9th International SDL Forum, pp. 225-240, June 1999.

[19] H. Behrens, "Requirements analysis using statecharts and generated scenarios," In: Doctoral Symposium at IEEE Joint Conference on Requirements Engineering, Sep. 2002.

[20] T. Maier and A. Zundorf, "The Fujaba statechart synthesis approach," In Workshop on Scenarios and State Machines: Models, Algorithms and Tools (SCESM '03), May 2003.

[21] S. Uchitel, J. Kramer, and J. Magee, "Synthesis of behavioral model from scenarios," IEEE tran. SE, vol. 29, no. 2, pp. 99-115, Feb. 2003.

[22] M. Glinz, "Systematically combining specifications of internal and external system behavioral using statecharts," Proc. 3'rd Int. Workshop on Scenarios and State machines: Models, Algorithms and Tools, pp. 14-20, May 2004.

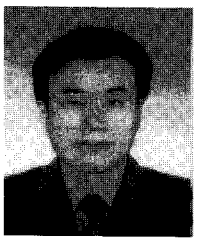
[23] D. Harel, H. Kugler, and A. Pnueli, "Synthesis revisited: generating statechart model from scenario-based requirements," Formal Methods in Software and Systems Modeling, LNCS 3393, pp. 309-324, 2005.

[24] B. Ludemann, "Synthesis of human-readable statecharts from sequence diagrams in ROOM environment," Diploma Thesis, Christian Albrechts Universität zu KielAug., Aug. 2005.

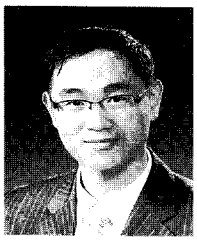
[25] R. Eshuis, "Statecharting petri nets," Beta Working Paper WP-143, Eindhoven

University of Technology, 2005.
[26] R. Eshuis, "Reconciling statechart semantics," Science of Computer Programming, vol. 74, no. 3, pp. 65-99, Jan. 2009.
[27] 장연세, UML기반 시스템 분석 설계, 이한출판사, 2008년 5월.
[28] D. Latella, I. Majzik, and M. Massink, "Towards a formal operational semantics of UML statechart diagrams," Proc. 3'rd Int. Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS99), pp. 331-347, Feb. 1999.
[29] Z. Pap, I. Majzik, A. Pataricza, and A. Szegi, "Methods of checking general safety criteria in UML statechart specification," Reliability Engineering & System Safety, vol. 87, no. 1, pp. 89-107, Jan. 2005.
[30] Z. Pap, I. Majzik, A. Pataricza, and A. Szegi, "Completeness and consistency analysis of UML statechart specifications," Proc. IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop (DDECS' 2001), pp 83-90, Apr. 2001.
[31] A. Karatkevich, "Deadlock analysis in statecharts," Proc. Forum on specification & Design Languages, Forum on specification on Design Languages, pp. 414-424, Sep. 2003.
[32] IPA Security Policy, "Toshiba Solutions Cryptographic Library V.1.0.1," Toshiba Solutions Corporation, Mar. 2007.
[33] S. Marquess, "OpenSSL FIPS 140-2 Security policy V.1.1.1b," Open Source Software Institute, Jan. 2007.
[34] StarUML 프로젝트, http://staruml.sourceforge.net/ko/index.php

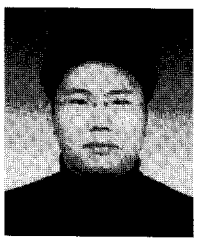
〈著者紹介〉



이 강 수 (Gang-Soo Lee) 종신회원
1981년 2월: 홍익대학교 전자계산학과 학사
1983년 2월: 서울대학교 대학원 전산학과 석사
1989년 : 서울대학교 대학원 전산학과 박사
1985년 ~ 1987년 : 국립한밭대학교 전자계산학과 전임강사
1992년 ~ 1993년 : 미국일리노이대학교 객원교수
1995년 : 한국전자통신연구원 초빙연구원
1998년 ~ 1999년 : 한남대학교 멀티미디어학부장
1987년 ~ 현재 : 한남대학교 컴퓨터공학과 정교수
〈관심분야〉 소프트웨어공학, 병행시스템 모델링 및 분석, 보안공학, 정보보호시스템 평가, 멀티미디어교육 커리큘럼



정 재 구(Jae-Goo Jeong) 정회원
2008년 2월: 한남대학교 학사
2008년 3월 ~ 현재: 한남대학교 컴퓨터공학과 석사과정
〈관심분야〉 소프트웨어공학, 보안공학, 정보보호 컨설팅 및 위험분석



고 갑 승(Kab-Seung Kou) 정회원
2005년 2월: 영동대학교 컴퓨터공학과 학사
2007년 2월: 한남대학교 대학원 컴퓨터공학과 석사
2007년 3월 ~ 현재: 한남대학교 대학원 컴퓨터공학과 박사과정
〈관심분야〉 소프트웨어공학, 보안공학, 정보보호 컨설팅 및 위험분석