자바 카드 애플릿의 검증 방법에 관한 연구

윤치영*, 염희균*, 전성익**, 황선명*
*대전대학교 컴퓨터공학과
**한국전자통신연구원
e-mail: mrmeg@zeus.dju.ac.kr

A Study on Verification Method for Java Card Applet

Chi-Young Yoon*, Hee-Gyun Yeom*
Sung-Ik Jun**, Sun-Myung Hwang*
*Dept of Computer Engineering, DaeJeon University
**ETRI

요약

자바카드 상에 탑재된 응용 애플릿의 품질을 측정하는 것은 중요하다. 이는 애플릿 테스트 베드를 구축하기 위해 필수적인 과정으로, 본 연구에서는 자바 카드용으로 개발되는 응용 애플릿의 품질을 분석하고 검증하는 방법이 필요하다. 애플릿과 호스트간의 커뮤니케이션 시에 발생되는 이벤트들을 케이스별로 구분하여, 자바카드 런타임 환경(JCRE)을 통하여 작동되는 Command APDU 와 Response APDU들의 커뮤니케이션 테스트 케이스를 검증하는데 정형 검증 방법 이용이 가능하다.

1. 서론

스마트 카드는 오늘날 사용되는 가장 작은 컴퓨팅 플랫폼 중 하나로 나타나고 있다. 자바카드의 기술적 설계에 대한 위대한 도전은 애플리케이션을 위한 충분한 영역을 보존하는 스마트 카드내의 적합한 자바 시스템 소프트웨어를 만드는 것이다. 그 해결책은 자바 언어의 특징에 해당하는 부분만을 지원하고 자바 가상머신(JVM)을 구현하기 위한 분산 모델에 응용하는 것이다[1]. 자바 카드 가상머신(JCVM)은 두 파트로 나뉘는데 카드 상에서 동작하는 것과 그렇지 않은 것이다. 런-타임에서 수행되기 위한 제약을 갖지 않는 많은 업무들에는 클래스 로딩, 바이트 코드 검증, 해법과 링크 최적화 등이 있는데 오프-카드 상에서 동작되는 가상기계에 한정되는 것들이다[1].

자바 카드 기술은 근본적으로 자바프로그램 언어로 작성된 애플리케이션이 스마트카드 상에서 작동 될 수 있는 플랫폼을 정의하는 것인데 미래의 자바 카드 플랫폼을 위해 작성된 애플리케이션을 애플릿이라 부른다.

자바 카드란 COS(Card Operating System)위에 JCVM이 랩핑(Wrapping)되어 있는 구조의 스마트 카드를 말하며, 자바 카드 애플릿이란 자바 카드 플랫폼에 맞게 설계된 자바 카드용 응용 프로그램을 말한다. 그러므로 자바 카드 애플릿의 구현에는 기존의 자바 환경에서 개발되는 자바 응용 프로그램 구현 조건과는 다른, 카드가 가지는 제한적인 하드웨어 리소스 활용 방안과 CAD(Card Acceptance Device)와의 커뮤니케이션 인터페이스, JCVM, 그리고 JCRE(Java Card Runtime Environment) 요구조건과 같은 카드의 소프트웨어적인 특성, 그리고 이러한 특성을 효과적으로 검증하기 위한 검증 방안 등이 고려되어야 한다.

본 논문의 구성은 2장에서는 관련 연구로 자바 카드 애플릿의 정의 및 특징과 구조에 대해 간단히 살펴보고, 3장에서는 정형 기법을 이용한 애플릿 커뮤니케이션의 검증 방법에 대해 설명하고, 4장에서 결론을 맺는다.

2. 자바 카드 애플릿

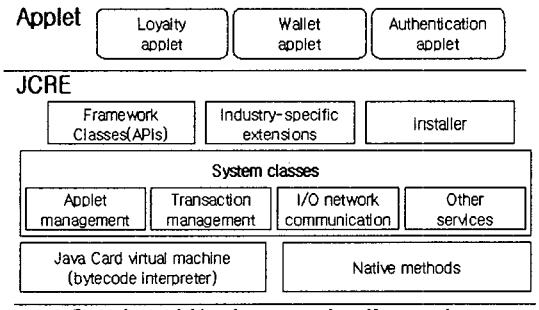
자바 카드의 애플릿이란 자바 언어로 작성된 카드 응용 프로그램을 말하며, JCVM에 의해 해석될 수 있는 바이트 코드 형태로 개발된다.

2.1 자바 카드 런타임 환경(JCRE)

JCRE는 스마트 카드 내부에서 작동되는 자바카드 시스템 컴포넌트들로 구성된다. JCRE는 카드 자원 관리, 네트워크 통신, 애플릿 실행, 카드 시스템과 애플릿 보안등에 관한 역할을 맡는다. 그러므로 스마트 카드 OS로서의 필수적인 서비스를 제공한다[5].

JCRE는 그림 1에 나타낸 바와 같이 JCVM, 네이티브 메소드, 시스템 클래스, API 그리고 인스톨러 영역으로 구성된다[4].

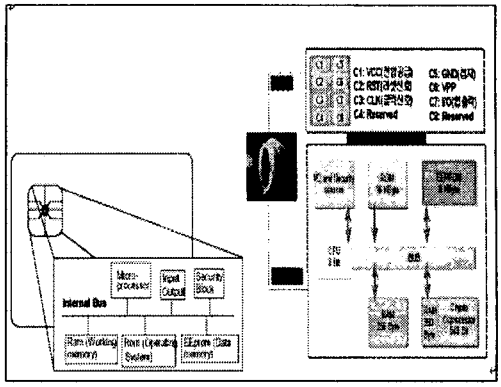
JCVM 영역에는 바이트 코드 해석을 위한 인터프리터가, 네이티브 메소드 영역에는 C 혹은 어셈블리로 구현된 모듈을 접근하기 위한 인터페이스가 구현되고, 시스템 클래스 영역에는 애플릿 관리, CAD와의 커뮤니케이션 및 트랜잭션 관리 그리고 시스템과 애플릿의 시큐리티 관리 기능이 구현되며, API 영역에는 자바 카드 API와 Industry Specific API 코드가, 그리고 인스톨러 영역에는 애플릿 다운로드 엔진이 구현된다.



<그림 1> 자바카드 런타임 환경(JCRE) 구조

2.2 자바 카드의 구조

자바 카드의 소프트웨어 구조는 그림 3에 주어진 바와 같이 COS, JCVM, 자바 카드 API, Industry Specific API, 그리고 애플릿으로 구성된다. COS 영역에는 메모리 액세스 및 I/O 핸들링을 위한 디바이스 드라이버와 암호 모듈 액세스 드라이버 코드가, JCVM 영역에는 자바 바이트 코드 서브셋 지원을 위한 자바 인터프리터 코드와 서명과 로그-인 같은 외부 접근 통제 코드가 적재된다. 자바 카드 API 영역에는 자바 카드 구현을 위한 기본 API 코드, Industry specific API 영역에는 사용자에 의해 정의되는 암호 API 코드가 적재되며, 애플릿 영역에는 전자화폐, 로열티, 그리고 신분 증명 등과 같은 응용 프로그램이 적재된다[4].



<그림 2> 자바카드의 구조

2.3 정형 기법 (Formal Method)의 개요

시스템의 개발에 있어서 그 신뢰도를 향상시키기 위해 많은 테스트가 필요하다. 그러나 시스템 개발 단계에서 테스트가 소비하는 시간, 인력, 비용은 무시할 수 없다[4]. 특히 시스템이 대형화, 복잡화되면서 테스트만으로는 그 신뢰도를 향상시키는 것이 불가능하게 되었다[7]. 또한 네트워크 환경이 발달하면서 시스템들이 통합되어 있지 않고, 통신망을 이용해서 분산환경을 구축하여 독자적으로 동작하는 분산 시스템이 개발됨에 따라 시스템의 신뢰도 검사가 더욱 어려워지고 있다[7]. 이러한 문제를 해결하고 시스템의 신뢰도를 향상시키기 위해 시스템의 설계시 정확한 명세와 검증이 더욱 필요하게 되었고, 이런 요구사항을 만족시키기 위해 정형기법을 사용한다[5][6].

정형 기법은 수학과 논리학을 기초로 하여 시스템을 수학적인 객체로 취급한다. 이러한 객체의 성질과 동작을 묘사하고 예측하기 위해 수학적 모델링을 수행한다. 이런 모델에 대해서 수학적인 기호를 사용하여 명세하기 때문에 시스템의 명세 시 자연어가 발생할 수 있는 애매 모호함이나 불확실성을 최소한으로 줄일 수 있다[3].

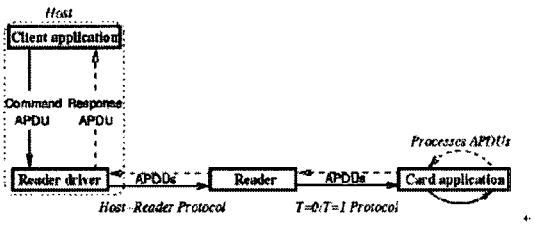
수학적인 성질을 이용해 시스템이 처음에 설계한 요구사항과 동일한지를 증명할 수 있다. 이러한 정형 기법을 사용하면 시스템을 구현하기 이전에 그 기능의 정확성을 검사할 수 있어 개발비용과 시간을 줄일 수 있다[3]. 특히, 분산 시스템의 경우 테스트에서는 발견하기 어려운 동작 환경이나 발생 가능한 오류를 모델링 하고 검사할 수 있어 그 신뢰도를 더욱 향상시킬 수 있다. 정형 기법은 크게 정형 명세와 정형 검증으로 구분된다[2].

2.4 Z method

정형 명세 언어로 표준 집합론에 기반하고 시스템의 특성을 표현하기 위해서 일차 논리(First-Order Logic)와 집합론(Set Theory)과 같은 수학적 개념을 확장시킨 정형 표기법이다. 수학적인 기반을 가지고 있기 때문에 명세에 많은 이득을 가지고 있다.
Z에서 구조를 구성하는 기본이 스키마이다. 스키마는 선언부(declaration)와 기술부(predicate) 두 부분으로 나누어진다. 선언 부분은 w:Type 형태의 변수 선언을 구성한다. 변수 이름은 w, Type은 스키마에 사용되기 위해 선언된 기본 타입(basic type) 이름이다. 기술 부분은 선언된 변수사이에 관계성을 표현한다.

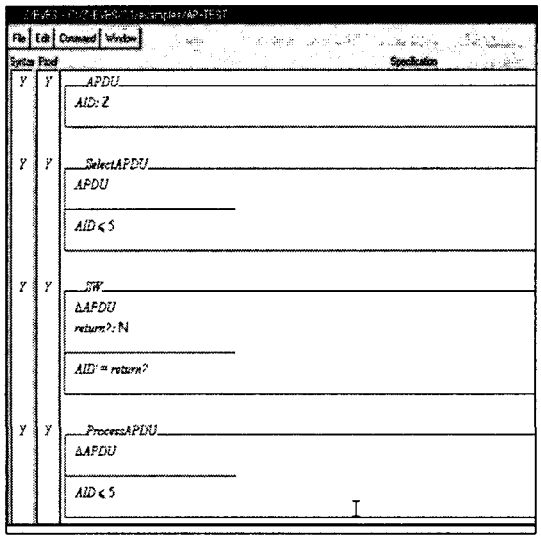
3. Z를 통한 애플릿 커뮤니케이션 검증

자바 카드 애플릿의 커뮤니케이션은 명령(Command) APDU와 응답(Response) APDU로 구성되는 APDU 교환을 통해 이루어지며 각 상태를 상태 다이어그램에서와 같이 확인 할 수있다. APDU 교환은 애플릿과 호스트간에 직접 이루어지는 것이 아니라 JCRE를 매개로 하여 이루어지고, JCRE는 애플릿과 호스트간에 교환되는 APDU의 관리 및 감독 역할을 담당한다[4][6].



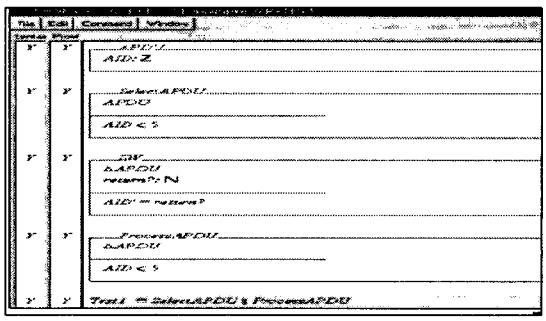
<그림 3> 자바카드 애플릿 커뮤니케이션

애플릿과 호스트간의 커뮤니케이션은 Command APDU 와 Response APDU로 구성되는 APDU 교환을 통해 이루어진다. APDU 교환은 애플릿과 호스트간에 직접 이루어지는 것이 아니라 JCRE를 매개로 하여 이루어지고, JCRE는 애플릿과 호스트간에 교환되는 APDU의 관리 및 감독 역할을 담당한다. 따라서 애플릿과 CAD 또는 호스트간의 직접적인 인터액트(Interact)는 불가능하며, JCRE를 통한 인터액트만이 가능하다. 여기서는 간단히 각각에 대한 상태들을 정형 명세화 할 수 있다.
자바 카드 애플릿에서 사용되어지는 4개의 필수 메소드를 Z를 이용하여 스키마 작업을 하고[7], 스키마를 자동화된 도구를 이용하여 검증한 결과는 다음과 같다.



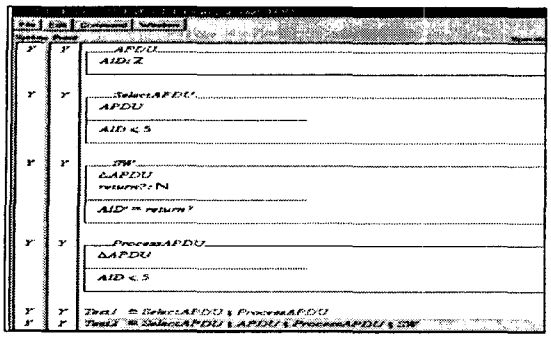
<그림 4> 애플릿 커뮤니케이션 명세

3.1 애플릿 커뮤니케이션 테스트 케이스 4가지 검증
① Case 1 : Select method를 수행한 후, 애플릿이 미리 선택되었는지를 확인한 후 원하는 Process method를 수행하게 되는 경우이다.



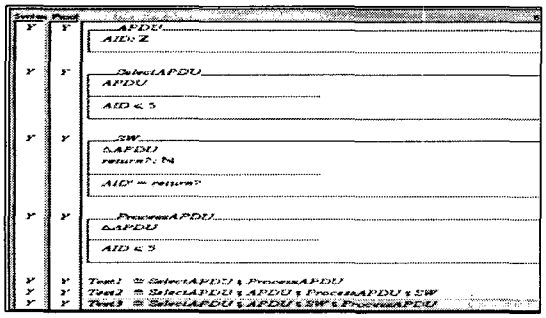
<그림 5> Case 1 검증

② Case 2 : Select method 명령어를 수행한 후, 애플릿 AID를 확인하여서 그 애플릿이 이미 선택되어졌다면 그 선택되어진 상태를 해제(deselect) method를 실행시킨 뒤, 다시 Select method를 수행한다. 그런 다음 Response APDU시 그 애플릿의 상태를 확인한 후에 요구한 호스트로 애플릿 상태를 Response하게 된다.



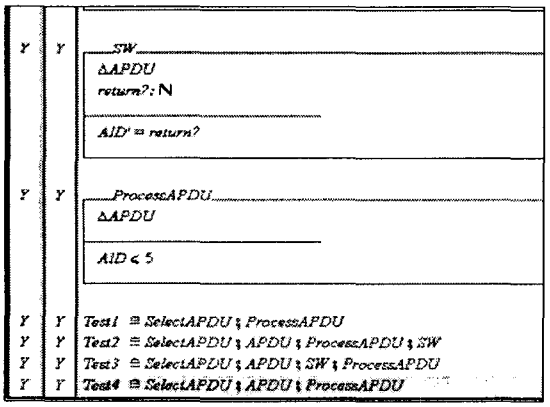
<그림 6> Case 2 검증

③ Case 3 : 애플릿을 사용하고자 할 때는 먼저 Select method로 시작하게 된다. 그런 다음 원하는 애플릿의 AID를 확인하고 Response APDU로 그 애플릿이 맞는지, 틀린지를 SW method를 이용하여 확인한 후에 Process method를 수행하게 되는 경우이다.



<그림 7> Case 3 검증

④ Case 4 : Select method로 시작한 후에 애플릿의 AID를 확인한 후에 바로 원하는 애플릿일 경우 바로 Process method를 수행하는 경우를 말할 수 있다.



<그림 8> Case 4 검증

4가지의 Case들을 정형 명세 한 이유는 어떠한 자바 카드 애플릿들은 위와 같은 4가지의 경우를 반드시 거친다는 점을 착안하여 이 4가지 Case를 정형 명세 한 후 이것이 정확히 명세 되었음을 검증한다면, 자바 카드 애플릿의 정확성 및 안전성을 구현 이전에 확인하고 검증할 수 있을 것이라는 것이다. 따라서 이러한 정형 기법을 이용하여 자바 카드 애플릿의 정확성 및 안전성을 검증 할 수 있다는 방법을 제시하고자 한다.

4. 결론 및 향후 연구

자바 카드 애플리케이션의 JCRE를 통한 애플릿 커뮤니케이션의 안전성 검증을 위해 Z 라는 정형 기법을 사용하여 정형 명세화 하고 작성된 정형 명세가 올바르게 작성되었는지를 검증하였다. 앞에서 설명한 것과 같이 이렇게 증명을 함으로써 작성된 명세가 문법적인 오류나 타입의 일관성에 대한 오류가 없다는 것을 Z/EVES라는 자동화 도구를 이용해서도 증명하였다.

향후 연구 방향으로는 정형 명세를 통하여 품질이 안정되고 요구된 기능을 가진 애플릿을 확인하고 등록된 애플릿의 호출에서부터 나타나는 여러 경우를 테스트하는 테스트 베드를 구축하며 검증 할 수 있는 연구 영역으로 확장하고자 한다.

참고문헌

[1] Joshua S., Judy S., "The Java Card Virtual Machine", JavaOne Conference, 1999.

[2] B. Ratcliff, 'Introducing specification using z : A Practical Case Study Approach', McGraw-Hill, 1994.

[3] J. M. Spiver, The Z notation: A Reference Manual, Prentice-Hall, 1989.

[4] Zhiqun Chen, Java Card Technology for Smart Cards : Architecture and Programmer's Guide, ADDISON-WESLEY, 2000.

[5] K. Lano, 'Formal Methods Fact File : VDM and Z', John Willey & Sons, pp.173-258, 1996.

[6] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic Specifications. ACM Transactions on Programming Languages and Systems, 1986.

[7] Mark Saaltink. The Z/EVES User's Guide. TR-97-5493-06, ORA Canada. Sep. 1997.