# 소프트웨어 상호운영성 시험 체계와 방법론

## (Framework and Methodology for Interoperability Testing of Software)

강 성 원 † 신 재 휘 †† 성 종 진 ††† 홍 경 표 ††††
(Sungwon Kang) (Jaehwi Shin) (Jongjin Seong) (Kyowngpyo Hong)

**요 약** 둘 혹은 그 이상의 개체가 협력하거나 통신할 때와 같이 상호작용할 때 그 상호작용의 올바른 정도를 상호운영성이라고 한다. 소프트웨어에서 상호운영성시험의 필요성은 복수의 소프트웨어 개체를 사용하여 주어진 기능 혹은 임무를 수행하는 시스템을 구축할 때 발생한다. 오늘날 소프트웨어가 더욱더 복잡한 문제를 해결하기 위하여 이용되고 소프트웨어시스템이 네트워크를 통하여 그 기능이 분산되어 더 풍부한 서비스를 제공하게 됨에 따라 상호운영성시험의 중요성은 더욱 더 커지고 있다. 이 논문은 상호운영과 상호운영성시험에 대한 근본적인 개념과 원리를 논의하고, 실제 상호운영성시험을 수행하기 위한 접근방법을 논의함으로써 상호운영성시험을 위한 체계와 방법론을 제공한다.

**키워드** : 소프트웨어시험, 프로토콜시험, 상호운영성, 상호운영성시험, 시험구조

***Abstract*** When two or more entities interact with each other as when they collaborate or communicate, the degree of correctness of the interactions is called interoperability. In software, the necessity of interoperability testing arises when we build a software system in which multiple interacting software entities are employed to perform the function or task of the system. These days as software is more and more being used to solve complex problems and through networking functions of software can be distributed to provide abundant services, the more the interoperability testing becomes important. This paper provides the framework and methodology for interoperability testing by discussing the fundamental concepts and principles that underlie interoperability and interoperability testing and the approaches for practicing interoperability testing.

**Key words** : software testing, protocol testing, interoperability, interoperability testing, test architecture

## 1. 서 론

적합성시험은 어떤 구현이 그 명세에 대하여 올바른가를 시험한다. 적합성시험은 이를 통하여 구현의 오류를 발견하고 명세에 의하여 정의된 구현들이 상호운영하기 위한 필요조건으로서 널리 수행되고 있다. 그러나 적합성시험은 두개의 중요한 제약이 있다. 첫째, 명세가 보통 불완전하거나 오류가 포함되어 있어 상호운영하여야 할 적합한 구현들도 서로 상호운영하지 않을 수 있다. 둘째로, 명세가 완전하고 무결하다고 하여도 시간, 메모리의 제한된 자원 그리고 적합성시험의 방법론적 한계에 의하여 보통 완전한 적합성 시험이 불가능하다. 따라서 상호운영하여야 할 구현들이 적합성시험을 통과하고도 실제에 있어서 상호운영하지 않는 경우가 자주 발생하게 된다. 따라서 서비스를 위하여 구현을 배치운영하기에 앞서 상호운영성시험을 수행하는 것은 필수적이다.

적합성시험의 방법론과 체계는 Conformance Testing Methodology and Framework(CTMF)로 불리우는 ITU-T X.290 Series 권고[1]로 표준화되어 주로 통신 프로토콜의 적합성시험을 위한 지침으로 세계적으로 널리 사용되고 있다. 또한 적합성시험을 위한 형식기법의 체계가 Framework on Formal Methods in Conformance Testing(FMCT)라 불리우는 권고로 ITU-T에서 표준화되었다[2]. 그러나 상호운영성시험에 대하여는

† 종신회원 : 한국정보통신대학교 조교수
kangsw@icu.ac.kr
†† 비 회 원 : 한국통신 운용시스템연구소 연구원
wiz@kt.co.kr
††† 비 회 원 : TTA IT시험연구소 팀장
jsung@tta.or.kr
†††† 비 회 원 : 한국통신 기술연구소 소프트스위치개발1실장
kphong@kt.co.kr
논문접수 : 2002년 10월 5일
심사완료 : 2003년 10월 9일

특정 통신프로토콜에 대한 상호운영성 시험스위트의 도출을 소개하거나[3,4] 상호운영성시험의 원리 및 상호운영성 시험스위트의 도출절차들에 대한 연구가 수행된 적은 있고[5], 적합성관계와 상호운영성관계의 형식적 정의 또는 적합성관계의 형식적 정의와의 관계에 대한 이론적인 연구는 있으나[6,7], CTMF에 상응하는 체계와 방법론은 현재 없는 상태이다. 예를 들어 논문 [4]은 CORBA ORB의 상호운영성 시험을 대상으로 구체적인 상호운영성시험의 경험을 시험구조의 결정 및 시험도출의 기술적인 측면에서 소개한다. 논문 [6]은 논문 [7]과 유사한 방향의 연구로, 형식적인 적합성관계에 대한 기존의 연구를 연장선상에서 형식적인 상호운영성정의를 제시하고 관련 성질들을 탐구한다. 본 논문에서 소개하는 상호운영성 시험체계와 방법론은 포괄적인 대상을 위한 것이며, 상호운영성시험에 관계되는 개념적인 기반 및 절차적 기반을 제공한다. 상기한 연구내용은 본 논문에서 제시하는 상호운영성시험 체계의 구성요소로 간주될 수 있다.

이 논문은 이와 같은 공백을 메우고 상호운영시험의 수행자들이 상호운영시험의 원리를 이해하고 실제 시험에 지침으로써 활용할 수 있는 상호운영성체계와 방법론 Interoperability Testing Framework and Methodology(ITFM)을 제시하는데 그 목적이 있다. 이를 위하여 제2절에서는 상호운영과 상호운영성시험의 기본개념에 대하여 논의하고, 제3절에서는 상호운영성시험의 체계, 제4절에서는 상호운영성시험의 방법론을 논의한다. 본 논문의 ITFM은 가능한 한 CTMF와 일관되도록 작성되었다[1].

## 2. 상호운영성과 상호운영성시험

### 2.1 상호운영성

두개 이상의 개체(entity)가 어떤 기능 혹은 임무를 수행하기 위하여 정보를 교환 또는 공유하거나 협력할 때 그러한 능력을 상호운영성(interoperability)이라고 부른다. 분산응용과 통신프로토콜에서와 같이 상호운영성은 상호운영에 참여하는 개체들에 필수적인 성질로 간주된다. 여기서 (1) 둘 이상의 개체의 참여와 (2) 개체들이 함께 기대되는 대로 행위하는 것은 상호운영의 두 가지 본질적 특성이다. 따라서 둘 혹은 그 이상의 개체가 함께 기대되는 대로 행위 한다면 두개의 개체는 상호운영(interoperation)한다고 말할 수 있다.

이러한 상호운영과 개체의 개념에서 볼 때, 다양한 종류의 상호동작(interact)하는 행위를 상호운영의 일종으로 볼 수 있고 우리가 전체로서 보고자 하는 어떠한 대상도 하나의 개체로 간주할 수 있다. 예를 들어, 클라이언트-서버 구조의 응용에서 클라이언트와 서버, 협력프로세스(cooperating processes)에서의 프로세스, peer-to-peer 구조의 통신프로토콜의 peer, 통신프로토콜을 요청자-응답자(initiator-responder)로 볼 때, initiator와 responder 등이 각각 하나의 개체에 해당되고, 다시 이러한 개체를 상호운영하는 구성개체(subentity)들로 이루어진 시스템으로 본다면 구성개체들은 다시 각각 하나의 개체가 된다.[2]

### 2.2 상호운영성시험

상호운영 정도의 측정은 두 가지 차원에서 수행될 수 있다: 하나는 명세(specification) 차원이고, 다른 하나는 구현(implementation) 차원이다. 명세차원에서 상호운영의 측정을 상호운영성검증(interoperability validation)이라고 부르고 구현에 대하여 상호운영의 측정을 수행하는 활동을 상호운영성시험(interoperability testing)이라고 부른다. 상호운영성검증은 구현을 위한 명세 자체가 그 명세에 기반한 구현이 상호운영성을 가지도록 올바르게 되어 있는지를 측정한다. 구현에 대한 상호운영성의 측정은 서론에서 언급한 바와 같이 (1) 명세의 불완전성 또는 명세 상의 오류 또는 (2) 적합성시험의 불완전성으로 인하여, 명세의 검증과 적합성시험으로도 상호운영상의 문제를 완전히 해결하지 못하기 때문에 필요하다.

### 2.3 상호운영성시험 체계와 방법론

앞에서 내린 상호운영성의 정의는 상당히 일반적이다. 이 정의에 따르면 실세계(real world)의 다양한 개체들이 그 정의에 따라 상호운영하거나 또는 상호운영하지 않고 따라서 상호운영성시험의 대상이 될 수 있다. 그러나 이 논문의 관심 대상은 분산응용, 통신프로토콜, 상호협력하는 프로세스와 같은 소프트웨어를 대상으로 하고 이 논문의 주제인 상호운영성시험 체계와 방법론도 소프트웨어를 그 논의영역(universe of discourse)으로 한다.

체계(framework)란 근본적인 개념, 가정, 원리를 의미하고 방법론(methodology)은 체계 위에 서는 것으로서 접근방법, 방법, 기법과 같이 “어떻게 하는가”에 대한 논의를 의미한다. 상호운영성시험의 체계와 방법론이 주어지면 이를 준수하여 상호운영성시험방법의 개발자들이 상호운영성시험을 위한 구체적 방법을 고안하고, 상호운영성시험 수행자들은 시험을 위하여 적절한 상호운영성 시험구조를 선택하고, 상호운영성 시험스위트를

1) 본 논문의 많은 부분은 현재 ITU-T SG17에서 X.itfm의 문서번호를 부여 받아 표준화 작업이 진행 중이다.

2) 이 논문에서 말하는 개체는 실세계의 개체일 수도 있고, 실세계의 문제를 해결하는 소프트웨어시스템의 개체일 수도 있다. 후자의 경우 개체는 연산개체(computing entity), 저장개체(storage entity), 통신채널 등일 수도 있고 혹은 이러한 것들의 복합체일 수도 있다.

도출하고, 상호운영성 시험절차를 실행하고 최종적으로 시험결과를 분석하고 보고할 수 있다.

방법(method)이란 어떤 것을 실행하는 구체적인 접근방법, 절차를 말한다. 하나의 상호운영성 시험방법은 상호운영성시험을 수행하는 구체적인 방법이다. 따라서 상호운영성시험 방법은 상호운영성시험을 실행하는 구체적인 공정, 절차를 말하며, 시험카버리지의 결정, 시험비용의 할당, 정적 상호운영성 검토, 상호운영성 시험구조의 결정, 상호운영성 시험스위트의 도출, 동적 상호운영성시험과 판정할당등을 포함한다[3].

방법과 대조적으로 방법론(methodology)은 방법에 관한 논의이다. 따라서 어떤 것을 달성하기 위한 기계적이고 단계적인 길을 제시하지 않는다. 방법론은 방법의 사용자에게 많은 선택의 여지를 남긴다. 방법의 사용자가 접근방법을 선택하거나 발명하는데 있어서 창조적이어야 할 수도 있다. 방법에 관한 공통적 측면과 필수적인 측면들을 제시할 수 있다. 반면에 체계는 방법론 또는 방법론이 제공하는 지식에 기반하여 특정한 방법을 고안하고 적용할 수 있는 발판을 제공한다.

## 3. 상호운영성시험 체계

### 3.1 접속, 개체, 연관

앞에서 실세계는 개체들을 포함한다고 말하였다. 그러나 이 개체들은 고립된 개체로서 존재할 수도 있지만 어떤 개체들은 서로 관련(association)지워져 있고 그들간에 어떤 상호작용이 있다. 이 상호작용의 방식을 접속(interface)이라고 부르기로 한다.

소프트웨어시스템도 마찬가지로 서로 관련지워져 상호작용하는 개체들로 구성된다. 보통 소프트웨어 시스템에서의 개체들은 서로 고립되어 있지 않고 상호작용을 통하여 궁극적인 시스템기능 혹은 시스템의 서비스를 창출해 낸다. 이 개체들은 그 안에 더 세부적인 조직을 가진 구성시스템(subsystem)일 수도 있고 더 이상 구성시스템으로 볼 수 없는 원자적 개체일 수도 있다. 소프트웨어 개체들도 실세계와 마찬가지로 접속을 통하여 상호작용하게 된다.

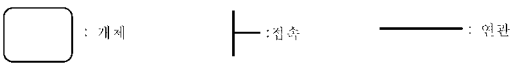



그림 1 실세계 혹은 소프트웨어시스템의 구조를 나타내기 위한 기본 구성체

3) 상호운영성시험의 이러한 사항들은 공정에서 특정한 순서로 발생하게 된다. 만일 충분히 구체적으로 개발되어 있는 특정한 방법을 따르게 되면, 시험수행자는 상호운영성시험을기계적으 처음부터 끝까지 완수할 수 있게 된다. 따라서 시험수행자는 하나의 단계가 끝날 때마다 다음에 어떤 사항을 어떻게 수행하는지에 대하여 창조적이어야 할 필요가 없다.

앞 절에서 제시된 실세계 혹은 소프트웨어시스템에 대한 관점(view)은 개체의 내부(internals)에 대하여 언급하지 않고 단지 개체간의 외적관계(external relation)에만 초점을 맞추도록 하여 실세계와 소프트웨어시스템을 구조적인 관점(architectural viewpoint)에서 볼 수 있도록 한다. 이 논문의 상호운영성에 대한 논의를 엄격하고 용이하게 할 수 있도록, 실세계 혹은 소프트웨어시스템를 구조적인 관점에서 나타내기 위하여 그림 1의 기본 구성체(construct)들을 사용한다. 이 기호들은 (1) 실세계의 구조, (2) 소프트웨어시스템의 구조뿐 아니라 (3) 소프트웨어 시험시스템의 구조를 나타내기 위하여 사용될 수 있다.

하나의 개체는 하나 혹은 그 이상의 접속을 갖는다. 접속은 개체가 외부세계와 상호작용하기 위한 “창문(window)”이다. 접속은 두 개의 개체간의 관계(binary relation)로 정확히 두개의 개체가 하나의 접속을 공유한다. 접속이 없는 개체는 정의상 다른 어떤 개체도 그것과 상호작용할 수 없고 그러므로 그것에 어떠한 정보를 주지도 그것으로부터 끌어내지도 못하므로 흥미를 주지 못한다.

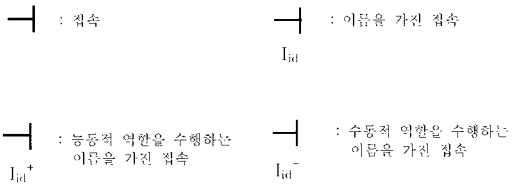



그림 2 접속을 나타내는 기호

각 접속은 그 유형(type)과 역할(role)을 가진다. 개체는 속성, 함수, 이벤트와 같은 접속요소(interface element)를 통하여 상호작용을 하며 각 객체는 각각의 접속요소에 대하여 initiator 혹은 responder의 역할을 가질 수 있다. 접속요소와 역할의 조합(combination)은 접속의 유형을 결정짓는다. 만일 하나의 접속에서 접속요소의 역할이 일관되게 initiator혹은 responder라면 “+” 혹은 “-”의 기호를 써서 편리하게 접속의 역할을 나타낼 수 있다. 즉 접속하는 두 개체는 그 접속에 있어서 반대의 역할을 하여야 한다. 좀 더 정확히 말하면, 두 개체의 상호작용에 사용되는 각 접속요소(interface element)에 대하여 그 반대 역할을 하여야 한다. 그림 2는 접속에 그 이름을 준 경우, 그 역할까지 표시한 경우의 접속기호를 보여준다.

연관은 두 개의 개체가 서로로부터 고립된 있지 않음을 나타낸다. A와 B가 연관되어 있으면, A가 B를 알거나(따라서 사용자가 되거나), B가 A를 알거나, A와 B

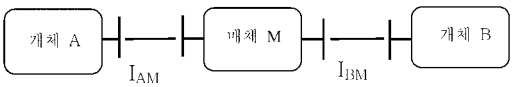


그림 3 객체 A와 B가 매체 M을 통하여 연관된 구조

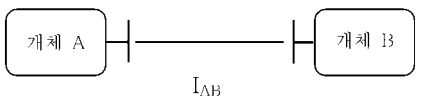


그림 4 객체 A와 B가 연관된 구조

가 서로 안다. 연관은 물리적인 정보매체가 아니고 추상적인 관계일 뿐이다. 따라서 개체 A, B가 매체 M을 통하여 연관되어 있음을 나타내기 위하여 그림 3과 같이 나타낼 수 있다. 많은 경우 우리는 개체 A, B가 연관되어 있다는 데에만 관심이 있고 그 연관이 어떻게 물리적으로 구현되어 있는가에 관심이 없을 수 있다. 이와 같이 그림 3에서 매체 M으로부터 추상하고 싶다면 그림 4와 같이 나타낼 수 있다.

### 3.2 상호운영구조

이 논문의 상호운영성시험 체계와 방법론의 전개를 위하여, 다음과 같은 가정을 하고 상호운영성시험의 여러 가지 필수적인 측면에 집중하고자 한다:

A1) 상호운영성시험의 대상이 되는 객체의 수는 2개이다.

A2) 한 객체가 갖는 접속의 수는 최대 2개이다.

그러나, 이 논문의 체계와 방법론이 적용될 경우 이 가정은 제거될 수 있고 체계와 방법론은 3개 이상의 개체들로 어려움 없이 일반화 될 수 있다.

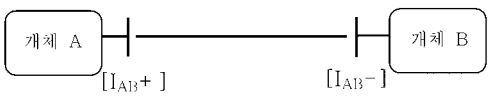


그림 5 두 개의 상호작용하는 개체

상호운영성의 정의와 앞의 가정에 따르면, 가장 단순한 종류의 상호운영의 경우는 그림 5와 같다. 개체 A와 개체 B가 단 하나의 접속(즉 $I_{AB}$)을 통하여 서로 상호작용할 수 있다.

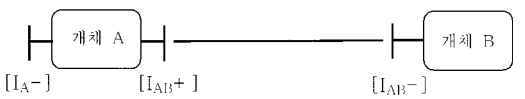


그림 6 클라이언트-서버 구조 예

약간 더 복잡한 경우가 개체 A 혹은 개체 B가 둘 이상의(즉 앞의 가정 A2에 의하여 두개의) 접속을 가진 경우이다. 그림 6은 그러한 경우를 그린다. 그림 4에서 개체 A는 두개의 접속을 가지고 있고 $I_{AB}$ 뿐 아니라 두 번째 접속 $I_A$를 통하여 상호작용할 수 있다. 그림 6의 구조는 클라이언트-서버 구조의 응용에서 찾아볼 수 있는데 이 경우 개체 A가 클라이언트, 개체 B가 서버이고 $I_A$가 사용자와의 그래픽 접속으로 볼 수 있다[4].

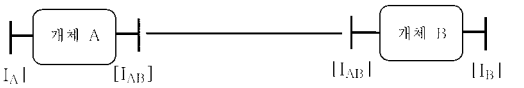


그림 7 Peer-to-peer 구조의 예

그림 7에서 개체 B는 추가적으로 접속 $I_B$를 가지고 있다. 그림 7이 모형화하는 상황은 개체 A와 개체 B가 통신 프로토콜(혹은 통신 프로토콜스택)의 peer인 경우 발생할 수 있다. 그러한 경우, 개체 A와 개체 B가 구현하는 통신 프로토콜은 개체 A와 개체 B의 사용자들에게 통신서비스를 제공하는 것으로 생각될 수 있다. 접속 $I_A$와 $I_B$는 각각 Service Access Point (SAP)라고 불리울 수 있다. $I_A$와 $I_B$는 동일한 유형의 SAP의 두 인스턴스(instance)일 수 있다. 만일 개체 A 와 개체 B가 클라이언트-서버 구조 혹은 master-slave 구조와 관련지워 진다면, $I_A$와 $I_B$는 서로 다른 SAP일 수 있다.

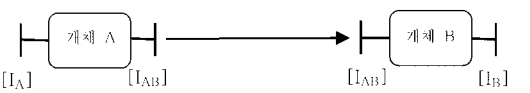


그림 8 Master-slave 구조 예

그림 8의 사례는 master-slave 구조에서 볼 수 있다. 연관관계가 일방적인 경우를 화살표로 나타내고 있다. 즉 A는 B를 알지만 B는 A를 알지 못하는 관계이다. B는 A로부터 명령을 받는 관계이고 A는 master, B는 slave가 된다.

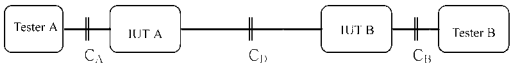
요약하면, 이 절에서는 A1), A2)의 가정 하에서 생성될 수 있는 상호운영구조들을 살펴보았고 그 상호운영구조의 사례들은 소프트웨어시스템의 구조에서 흔히 찾아볼 수 있는 것이다.

### 3.3 상호운영시험구조

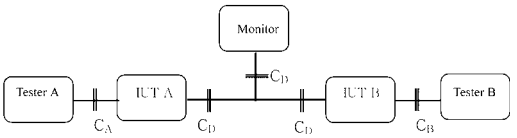
시험기(tester)는 (완전한) 시험 시스템의 구성요소이다. 시험기에는 두 가지 종류가 있다: (1) 능동적 시험기와 (2) 수동적 시험기이다. 능동적 시험기를 "시험기" 그리고 수동적 시험기를 "모니터"라고 부른다. 모니터는 단지 관찰하므로 상호작용에는 영향을 주지 않지만 (능

4) 개체 A와 개체 B는 그 자체가 복합개체일 수 있고 이 경우 구조관점에서 더 자세히 기술될 수도 있다. 클라이언트-서버구조를 클라이언트개체와 서버개체를 더 상세하게 보여주는 예가 [5]에 있다.

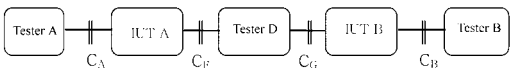
동적) 시험기는 관찰할 수도 있고 상호작용을 간섭할 수도 있다. 각각의 시험기는 대상 시스템의 행위에 대하여 그것이 보는 관점에서 올바름에 대하여 판단을 내릴 수 있다. 시험의 대상이 되는 소프트웨어개체를 시험대상구현(IUT: Implementation Under Test)이라고 부른다.




(a) 상호운용성 시험구조 I




(b) 상호운용성 시험구조 II




(c) 상호운성성 시험구조 III

그림 9 peer-to-peer시스템을 위한 세계의 상호운영성 시험 구조s

실제시스템은 다양한 구조로 존재하고 시험이 지속되는 동안 그 동작이 제어/관찰될 수 있는 방식이 다양하다. 그러므로, 다양한 상호운영성 시험구조(interoperability test architecture)가 적용될 수 있다. 3.2절의 가정 A1, A2 하에서, 3.2절에서 소개된 4개의 상호운영기본구조를 위한 시험구조를 고려할 경우 여러 가지 다양한 시험구조가 가능함을 알 수 있다. 예를 들어 그림 5의 시스템구조를 위한 여러 가지 상호운영성시험 구조 가운데 특히 그림 9의 3개를 생각할 수 있다. 그림 7에서 (a)는 2개의 시험기를 사용한 상호운용성 시험구조, (b)는 2개의 시험기와 모니터를 사용한 상호운용성 시험구조, (c)는 3개의 시험기를 사용한 상호운용성 시험구조이다.

세 개의 구조는 두 개의 IUT 사이의 상호작용을 제어하고 관찰하는 서로 다른 능력을 갖고 있다. 구조 I은 IUT A와 IUT B사이의 상호작용에 대한 어떠한 관찰이나 제어도 수행하지 않는다. 반면에 구조 II는 상호작용을 단지 관찰하고, 구조 III은 상호작용에 대하여 관찰과 제어를 모두 수행한다. 구조 II에서 모니터는 상호작용을 관찰하지만 두 개의 IUT 사이의 상호작용에 영향을 주지 않는다. 따라서, 구조 II는 구조 I보다 더 강력하다. 구조 III에서 시험기 D는 두 개의 IUT 사이에 상호작용을 간섭(interference)할 수 있다. 예를 들어, 링크를 오가는 메시지를 가로챌 수 있고, 지연, 손실, 메시지의 반복을 시뮬레이션 할 수 있고 네트워크(network cloud)를 시뮬레이션 할 수도 있기 때문에, 두 개의 IUT 사이에 폭 넓은 범위의 시험시나리오를 수행할 수 있다. 따라서 구조 III은 가장 강력한 시험구조이고 다른 두 개의 구조가 갖는 능력을 모두 갖고 있다.

이와 같이 대상 시스템 구조에 초점을 맞추고 모든 관련 접속을 고려하여 다양한 시험 구조후보를 얻을 수 있다. 그러나 시험스위트 생성과 시험에 있어서, 추가적인 시험기는 시험시나리오 도출이나, 시험시스템 구현에 추가적인 비용을 요구하기 때문에 현실적으로 항상 가장 강력한 시험구조를 선택할 수 있는 것은 아니다.

따라서 시험을 위하여 시험카버리지와 시험비용 관점에서 다양한 후보로부터 목적과 자원에 맞추어 가장 적절한 시험구조를 선정할 필요가 있다. 일반적으로 같은 시험카버리지를 제공한다면, 시험 비용을 최소화 하기 위하여 시험되는 접속의 수를 최소화할 필요가 있다.

### 3.4 시험항목

하나의 상호운영성 시험항목(interoperability test case)은 시험목적과 시험시나리오를 갖는다. 시험목적은 시험항목이 검증하고자 하는 것을 말하며, 시험시나리오는 시험목적을 실제로 검증하기 위하여 시스템의 입출력을 제어/관찰하기 위한 일련의 상호작용이다. 이러한 목적으로 상호운영성시험항목의 시험시나리오는 시험전부(preamble), 시험본체(test body) 그리고 시험후부(post amble)의 세 개의 부분으로 이루어져 있다.

시험전부는 (상호운영)시스템, 즉 가정 A에 의하여 두 개의 IUT, IUT A와 IUT B를 시험본체가 시작될 수 있는 안정된 상태(stable state)[5]로 가져가는 부분이다. 시험본체는 IUT A와 IUT B로 이루어진 시스템이 시험항목의 목적에 부합하게 상호작용을 하는가를 확인하다. 시험후부는 흔히 비어있을 수도 있는데 IUT A와 IUT B가 시험본체가 시행된 후 안정된 상태에 도달하게 하는 부분이다.

### 3.5 시험스위트

상호운영성시험을 위한 시험항목의 집합을 상호운영성 시험스위트(interoperability test suite)라고 부른다. 하나의 시험항목을 도출하기 위하여는 시험구조가 먼저 확정되어 있어야 한다. 그러나 시험스위트의 모든 시험항목이 동일한 시험구조에서부터 도출되어야 하는 것은 아니다. 필요에 따라 복수개의 시험구조를 사용할 수 있고, 각 구조에 대하여 그로부터 도출된 시험항목의 집합이 있게 된다.

5) 안정상태에서 시험본체가 시작되지 않을 경우 일반적으로 시험본체가 그 전제조건에 맞는 상태에서 게 수행되어 시험본체의 실행결과에 대한 평가를 올바로 할 수 없다.

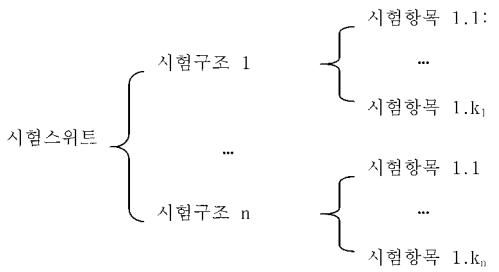



그림 10 시험스위트의 구조

### 3.6 시험실행과 판정할당

시험항목을 실행할 때 일반적으로 시험구조에 맞게 다양한 시험기가 병렬적으로 실행된다. 각 시험기는 자신의 관점에서 대상 시스템의 올바름(correctness)에 대하여 부분적평가(local evaluation)를 한다. 한 시험항목 전체에 대한 평가는 부분적 평가결과에 기초하게 된다. 시험항목의 실행이 끝날 때, 그 결과는 공식적으로 판정(verdict)으로 요약된다.

적합성시험에서도 여러 개의 시험기 사용될 수 있고 따라서 시험실행과 시험판정에 적합성시험과 상호운영성시험 사이의 근본적인 차이가 없다. CTMF와 같이 예비 판정값에는 none, pass, inconc, fail 있고 최종판정값은 pass, inconc, fail이다. 표 1은 예비판정이 새로운 판정값과 합성되어 어떤 값을 갖게 되는가를 보여준다.

표 1 판정의 계산

| 현재의 판정값 | 가장 최근 관찰된 행위에 대한 판정값 | | |
|---|---|---|---|
| | PASS | INCONC | FAIL |
| none | pass | inconc | fail |
| pass | pass | inconc | fail |
| inconc | inconc | inconc | fail |
| fail | fail | fail | fail |

적합성시험과 비교하여 상호운영성시험 시험 구조의 복잡도가 증가함에 따라 2개판정(즉, 현재의 판정과 가장 최근에 얻어진 판정)의 합성대신에 n개 판정의 합성을 고려할 필요가 있다. 또한 master 와 parallel 컴포넌트에서와 같은 2계층 시험시스템 구조가 아니라 n-계층 시험시스템 구조를 가질 수도 있다.

표 1을 2개판정의 합성을 보여 주고 있다. 표 1에 따르면 판정값은 다음과 같은 우선순위를 갖는다:

none < pass < inconc < fail

만일 v가 새로운 판정이라고 하면, lub가 least upper bound을 의미하고 $v_0$가 이전 판정이고 $v_i$, 1 <= i <= n,가 최근에 얻은 판정들의 집합일 때 n개 판정의 합성인 v는 다음과 같이 계산된다:

$$v = lub\{v_0, v_1, v_2, ..., v_n\}$$

## 4. 상호운영성시험 방법론

### 4.1 상호운영성 시험스위트 도출

앞에서 기술한 상호운영의 정의는 “기대되는 행위”만을 언급하고 그것이 무엇인지 자세히 기술하지 않았다는 의미에서 ‘열린’정의이다. 그 이유는 일반적으로 기대되는 상호작용이 시스템마다 다르기 때문이다. 그러므로, 상호운영을 위하여 요구되는 사항은 원칙적으로 구현의 명세를 통하여 명시적으로 주어지거나 명세작성자와 구현자 사이에 묵시적으로 동의되어야 한다. 또한 명시적인 분산응용이나 통신프로토콜의 상호운영성에서 보통 그러한 기대는 단일한 문서를 통하여 주어지지 않는다. 반면에 많은 경우에 관련된 명세로부터 유추되거나 도출되어야만 한다.

이러한 점을 고려하고 상호운영의 관점에서 기대되는 행위가 시스템의 명세(specification)에 충분히 상세히 정의되어 있다고 가정하자. 상호운영성시험스위트는 소프트웨어 개발의 여러 단계에서 생산되는 시스템명세, 설계명세, 구현과 같은 개발산출물(artifacts)들로부터 도출될 수 있다. 설계명세는 시스템구조와 시스템의 N개의 구성개체들에 대한 N개의 명세로 이루어진다고 볼 수 있다.[6] N개의 명세는 시스템 명세에서부터 설계결정에 의하여 구현을 개발하는 과정에서 얻어진 것일 수도 있고, 시스템 명세와 독립적으로 주어진 경우도 있다. 이 논문의 주제인 시스템시험과 인수시험의 관점에서 볼 때 상호운영성시험스위트의 도출 소스(source)로서 가장 중요한 두 개의 개발산출물은 시스템명세(system specification)와 설계명세(design specification)이고 여기서 구현프로그램을 제외된다[7].

그림 11에서 시험도출접근방법 D1은 시스템구조에서부터 하나이상의 시험구조를 선정하고 이를 기반으로 N개의 개체에 대한 N개의 명세에서부터 상호운영시험스위트를 도출한다. 이 시험스위트는 주어진 시스템구조하의 N개의 개체의 상호운영성시험을 위하여 적용된다. D2의 경우 구현은 비록 굵은 화살표와 같이 설계과정을 거쳐 이루어진다 하더라도, 원래의 시스템명세에서부터 시험스위트가 도출된다. 이 경우 시험구조는 전체시스템

6) 앞의 정의에 의하여 각각의 구성개체는 다시 다른 개체들로 구성된 내부구조를 가질 수 있다. 논점을 명확히 하기 위하여 여기서는 시스템과 구성개체들이라는 2계층의 구조만을 고려한다.

7) 구현프로그램을 상호운영시험스위트 도출의 소스로 보는 것도 불가능하지 않지만, 이 경우 소스코드 의존적인 시험스위트가 나와, 개발오류를 정정하는 능력이 떨어질 수 있고, 도출과정이 어려울 수 있는 단점이 있다.

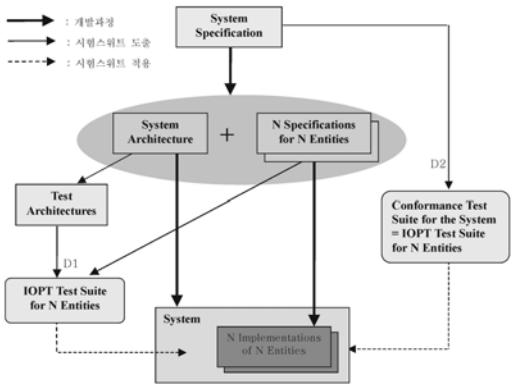



그림 11 상호운영성 시험도출을 위한 두 가지 접근방법

에 대한 적합성시험구조와 다를 바가 없다. 즉 시스템 내부의 어떤 구성개체가 존재하며 어떤 구조를 형성하고 있는지 알지 못하므로 시험스위트 도출에 반영시킬 수 없다. 따라서 이 시험스위트는 시스템에 대한 적합성 시험스위트이다. 다른 한편 이를 동시에 N개체들에 대한 상호운영시험스위트로 볼 수도 있다. 상호운영시험구조 가운데에 구성개체 간의 접속들을 무시하고 단지 구성개체와 시스템외부와의 접속만을 고려하는 구조를 생각할 수 있는데 시스템 적합성시험은 이와 구별되지 않는다[8]. 따라서 D2의 경우 N개의 개체의 상호운영성의 시험 문제를 시스템 적합성시험의 문제로 볼 수 있고, 적합성시험의 방법들을 적용할 수 있다.

최초의 명세가 시스템명세 형태로 주어지지 않고, 설계명세 형태로 주어졌다면, 그리고 시험도출을 위하여 설계명세로부터 특정구조를 가진 구조관점에서 내부를 들여다 볼 수 있는 시스템명세를 먼저 도출하였다면, N개체의 상호운영성시험을 위하여 역시 적합성시험 접근방법을 취하고 있는 것이다.

### 4.2 상호운영성시험 절차

이 절에서 설명하는 상호운영성 시험절차는 도출된 상호운영시험스위트가 존재한다고 가정한다.

상호운영성 평가 절차는 그림 12의 흐름도와 같이 요약될 수 있다. 시험수행자는 구현물의 제작자로부터 구현물의 구현적합성선어(ICS: Implementation Conformance Statement)와 시험을 위한 구현추가정보(IXIT: Implementation eXtra Information for Testing)을 제출받는다. (ICS와 IXIT의 정의에 대하여는 X.290의 제5.6절, 6.2절 참조 [1].) ICS와 IXIT에 기초하여 먼저 정적 상호운영성시험(검토)이 수행된다. 만일 검토결과가 fail이거나 동적상호운영성시험을 수행할 수 있을 만

8) 상호운영성시험과 적합성시험의 관계에 대하여는 추후 다른 곳에서 논의할 예정이다.

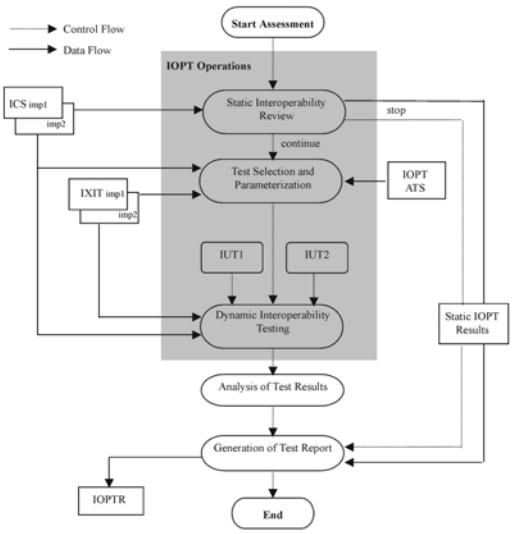



그림 12 상호운영성시험 절차

큼 구체적인 정보가 없다고 판단이 되면, 이러한 사항을 기록한 최종보고서를 만들고 평가는 종료된다. 그렇지 않을 경우 ICS와 IXIT의 정보에 기초하여 동적 상호운영성시험을 위한 시험항목들이 선정된다. 시험항목들은 (ICS에서 주장하는 공동의 명세을) 준수한다고 주장하는 IUT들에 대하여 선택된다. 이러한 선정단계가 필요한 이유는 일반적으로 상호운영성시험스위트는 모든 경우의 상호운영성시험을 위하여 준비된 것이므로, 특정한 경우에는 그 일부만이 정당히 적용될수 있는 시험항목들이기 때문이다. 최종적으로 상호운영성 시험항목은 실행되고 그 결과는 분석되어 시험의뢰자에게 보고된다.

## 5. 상호운영성 시험체계와 방법론의 준수

이 절은 특정한 상호운영성시험방법과 절차(이를 줄여서 상호운영성시험방법이라고 부르기로 한다)가 어떤 경우에 이 논문에서 제시된 상호운영성시험 체계와 방법론에 부합되고 혹은 부합되지 않는지에 대하여 논의한다. 특정한 상호운영성시험방법은 특정한 시험상황에 적용될 때, 그 방법의 강점과 한계가 나타날 수 있고 그 원인은 방법 자체에 기인할 수도 있고 특정 시험의 상황에 기인할 수도 있다. 본 절에서는 방법이 체계와 방법론에 부합하기 위한 조건을 논의하는 것이 목적이므로, 방법 자체의 관점에서 논의한다.

### 5.1 상호운영성

주어진 상호운영성시험방법에서 상호운영성의 정의를 “2개 이상의 개체들의 상호작용할 때 그 개체들간의 상호작용이 기대와 일치하는 정도”라고 정의하거나 혹은 이 정의의 필수적인 측면인 (1) 2개 이상의 개체들을

대상으로 한다는 것, (2) 명세와 같이 기대되는 상호작용이 실제로 올바른지 확인할 수 있는 근거에 비추어 상호운영의 정도를 가늠한다는 것을 내포하는 상호운영성 정의에 기초하여야 한다.

### 5.2 상호운영성시험

주어진 상호운영성시험방법에서 시험의 대상은 2개 이상의 개체 즉 IUT(Implementation Under Test)이어야 하고 시험스위트도출의 소스를 명확히 제시하여야 한다.

### 5.3 접속과 개체

주어진 상호운영성시험방법은 시험대상시스템을 개체, 접속, 연관으로 이루어진 결합체로 보아야 하며, 이들의 개념은 제3.1절에서 제시한 접속, 개체, 연관의 개념과 일치하여야 한다.

### 5.4 상호운영구조

주어진 상호운영성시험 방법은 개체, 접속, 연관의 개념 또는 기호로 명확히 표현된 시험대상시스템의 구조에서 출발하여 그 구조를 시험에 이용할 수 있어야 한다.

### 5.5 상호운영성시험구조

주어진 상호운영성시험 방법은 시험대상시스템의 구조로부터 시험카버리지와 시험비용을 고려하여 적절한 하나 이상의 상호운영성시험구조를 선정하고 각 상호운영성시험구조로부터 하나 이상의 상호운영성시험항목을 도출한다.

### 5.6 시험항목

주어진 상호운영성 시험방법을 사용하여 도출된 시험항목들은 시험전부, 시험본체 그리고 시험후부의 구조를 갖는다. 그리고 이 세 부분은 제 3절에 있는 정의에 부합된다.

### 5.7 시험실행과 판정 할당

주어진 상호운영성 시험방법은 다음의 사항을 요구한다:

(1) 각 시험항목의 실행은 시험기에 해당하는 컴포넌트들을 실행하여야 한다.

(2) 각 시험기는 부분평가를 수행하고 판정 대상 시스템의 올바름에 대하여 자신의 관점에서 판정을 준다.

(3) 시험항목실행에서 예비판정은 none, pass, inconc, fail의 4개의 값을 가질 수 있다. 시험항목실행의 최종판정은 합격(pass), 불합격(fail) 혹은 미확정(inconc)의 판정값을 갖는다.

(4) 판정계산은 제 3.6절처럼 이루어진다.

### 5.8 시험스위트 도출

주어진 상호운영성시험방법은:

(1) 그것의 시험스위트도출의 소스를 분명히 밝힌다. 특히 소스가 시스템명세인지 혹은 N개의 개체에 대한 N 개의 별개의 명세인지 분명히 한다.

(2) 상호운영성시험스위트는 순수한 상호운영성시험항목으로 구성되어 있는지 적합성시험항목을 포함하는지 밝혀야 하고, 후자의 경우 적합성시험항목에 대하여 그 시험항목이 어느 IUT의 적합성시험항목인지를 선언한다.

### 5.9 상호운영성 시험절차

주어진 상호운영성 시험방법은 다음의 사항을 요구한다:

(1) 각 IUT에 대하여 ICS와 IXIT를 요구한다.

(2) 정적 상호운영성시험을 수행한 후 검토결과가 불합격이거나, 동적상호운영성시험을 수행할 수 있을 만큼 구체적인 정보가 없다고 판단이 되면 이러한 사항을 기록한 최종보고서를 만들고 평가를 종료한다.

(3) 최종보고서에는 동적상호운영성시험을 수행하고 불합격 또는 미확정의 판정을 받은 각 시험항목에 대하여 그 원인을 기술하고 특히 그 원인이 명세에 있는지 구현에 있는지, 구현에 있을 경우 어는 구현에 오류가 있는지를 분명히 기술한다.

## 6. 결 론

지금까지 분산응용 소프트웨어와 통신프로토콜의 상호운영성시험을 위한 체계와 방법론에 대하여 논의하였다. 이 논문의 상호운영의 정의가 매우 광범위한 대상에 적용되듯이, 상호운영성시험은 매우 광범위하게 적용될 수 있는 시험이다. 그러나 지금까지는 한편으로는 통신프로토콜의 영역에서 다른 한편으로는 소프트웨어시험 중 통합시험이라는 분야로서 주로 연구되어 왔고 이 두 분야는 독립적으로 연구되어 왔고 상대분야의 연구결과를 수용하지 못하였다. 이 두 분야가 어떤 점에서 공통점이 있고 다르다면 어떤 점에서 다른지에 대한 이해도 부족하다.

이 논문에서는 이 두 분야를 같은 그릇에 담는 체계를 제시하였다. 그래서 이 논문의 체계와 방법론이 상호운영성시험 방법의 개발자나, 시험수행자를 위한 지침이 되고자 하는 원래의 목적 외에도, 이 두 분야가 발전하고 서로의 발전으로부터 상승효과를 볼 수 있는 발판 역할을 할 수 있기를 저자들은 희망한다. 또한 저자들은 이 논문의 체계와 방법론을 여러 실사례에 적용하여, 이 논문의 체계와 방법론을 준수하지 않는 방법과 비교할 때 준수하는 방법의 효과가 어느 정도인지 분석 할 계획이다.

## 참고문헌

[1] ITU-T Recommendations X.290 Series: Conformance Testing Methodology and Framework Parts 1-7.

[2] ITU-T Recommendation Z.500 Framework on formal methods in conformance testing, May 1997.
[3] ATM Forum BTD-TEST-INTR-01.09 Introduction to ATM Forum Test Specification, January 2001.
[4] M. Li, A. Puder, I. Schieferdecker, "A Test Framework for CORBA Interoperability," *5th. IEEE Int. Enterprise Distributed Object Computing Conf. (EDOC'01)*, Seattle, Washington, USA, Sep. 2001.
[5] Sungwon Kang, Jaehwi Shin, Myungchul Kim, "Interoperability Test Suite Derivation for Communication Protocols," *Computer Networks-The International Journal of Computer and Telecommunications Networking,* Vol. 32, pp. 347-364, March 2000.
[6] J. Aliovic-Curgus, S.T. Vuong, "A Framework for Interoperability Testing of Network Protocols," *Proc. Int'l Conf. on Network Protocols*, 1993.
[7] Sungwon Kang, "Interoperability and Conformance Relations of Communicating Systems," *Journal of the Korean Information Science Society (A): Computer Systems and Theory, Vol. A-24, No. 12*, December 1997.

강 성 원
1982년 2월 서울대학교 사회과학대학 졸업. 1989년 8월 미국 University of Iowa 전산학 석사. 1992년 12월 미국 University of Iowa 전산학 박사. 1993년 12월~2001년 10월 한국통신 연구개발본부 선임연구원. 2001년 10월~현재 한국정보통신대학교 조교수. 2002년 12월~현재 미국 Carnegie-Mellon University. 소프트웨어공학석사과정 겸임교수

신 재 휘
1991년 2월 한양대학교 전자공학 학사 1993년 2월 서울대학교 전자공학 석사 1996년 9월~1997년 8월 미국 NIST 객원연구원. 1993년 3월~현재 한국통신 운용시스템연구소 선임보연구원

성 종 진
1990년 2월 경북대학교 전자공학과 졸업(공학사). 1992년 2월 경북대학교 대학원 전자공학과 졸업(공학석사). 1992년 1월~2001년 11월 한국전자통신연구원 선임연구원. 2001년 12월~현재 TTA IT시험연구소 팀장

홍 경 표
1.1 한국항공대학 전자공학과 학사.
1.1 KAIST 전기 및 전자공학과 석사. 1997년 3월 KAIST 전기 및 전자공학과 박사. 1987년 3월~현재 KT 기술연구소 소프트스위치개발1실장. 2000년 5월~2001년 4월 UIUC CSL(Coordinated Science Laboratory) Visiting Researcher