論 文

객체 지향 모형을 이용한 TTCN 확장에 관한 연구

正會員 崔 重 珪* 正會員 宋 周 錫*

A Study on the Extension of TTCN using Object-Oriented Model

Joong Kyu CHOI* Joo Seok SONG* *Regular Members*

要 約 본 논문에서는 적합성 시험 표기 기법인 TTCN이 객체, 클래스, 상속 등의 객체지향 개념을 갖도록 확장하였다. 분산 시스템 환경하에서 ISDN과 같은 응용 프로토콜 구현제품들이 표준 프로토콜에 적합한 지를 시험하기 위한 시험 시스템은 구조가 복잡하고, 병렬로 수행하는 프로토콜 시험 행위를 포함하고 있기 때문에 기존의 TTCN으로는 전체 시험 suite를 이해하기 쉽고 명확하게 나타낼 수 없다. ISO에서는 TTCN에 병렬성을 표기할 수 있도록 확장하는 연구가 진행 중에 있으나, 이와 병행하여 병렬 시험 요소를 하나의 객체로 표기하는 객체 모형을 TTCN에 적용한다면 신뢰성과 소프트웨어 재사용을 증대시키는 효과를 얻을 수 있으며, TTCN을 하나의 시험 언어로 발전시킬 수 있을 것이다.

ABSTRACT This paper extends TTCN, a conformance test notation, using object oriented concepts including object, class, and inheritance. In distributed system environments, since test systems that should test the implementations for application protocols whether they conform to the standard protocol include protocol test behaviours executed in parallel, the conventional TTCN can't describe whole test suites explicitly. ISO is working the extension of TTCN including parallel notations, but concurrently if TTCN could be applied by the object model that regards a parallel test component as a object, we would gain the advantages of reliability and software reusability, and make TTCN a test language.

I. 서론

최근 정보통신의 급속한 발달로 네트워크를 통해 여러 지역으로 분산되어 있는 컴퓨터 시스템은 ISDN(Integrated Services Digital Network)과 같은 많은 응용 프로토콜을 이용하여 서로 통신하게 된다. 여러가지 프로토콜 구현제품들은 서로 상호작용을 하기 어렵기 때문에, 개방형 환경하에서 서로 호환을 이루도록 하기 위하여 ISO(International Organization for Standard)와 CCITT(International Telegraph and Telephone Consultative Committee) 등에서 정한 표준 프로토콜에 적합한 지의 여부를 시험해야 한다[1]. ISO와 CCITT에서는 적합성 시험 방법의 필요성을 인식하여 여러가지 표준안을 마련하고 있으며, 특히 ISO에서는 이러한 응용 프로토콜에 맞는 멀티파티시험 방법에 대한 연구가 진행 중에 있다[2]. 또한, ISO에서는 적합성 시험 표기 기법으로 TTCN(Tree and Tabular Combined Notation)을 제시하여 표준화하고 있으며[3][4], 멀티파티 시험 방법과 병행하여 TTCN에 병렬성을 부가하여 확장하는 연구가 활발하게 진행되고 있다[5].

확장되는 TTCN은 병렬로 수행하는 프로토콜 시험 행위를 포함하기 때문에 TTCN으로 작성된

*延世大學校 電算科學科
Dept. of Computer Science, Yonsei University
論文番號: 91-133 (接受1991. 9. 18)

시험 suite가 복잡하고 이해하기 쉽지 않다. 본 논문에서는 TTCN에 객체 지향개념을 도입하여, 병렬로 통신과 동기화를 수행해야하는 시험 요소들을 객체로 모형화하여 내부 프로토콜 행위를 캡슐화하고, 이들 엔터티간의 인터페이스를 추상화한다. 또한, 객체의 특성을 이용하여 병렬 시험 요소간의 병렬성을 지원하도록 하며, 시험요소 원형이라는 클래스를 정의하여 재사용이 가능한 시험 요소 원형간의 상속을 나타낸다. 따라서, 객체 지향 개념을 갖도록 TTCN을 확장하게 되면, 전체 시험 suite에 대한 이해도를 증대시키고, 시험 시스템에 대한 신뢰성 및 소프트웨어 재활용을 기대할 수 있다.

Ⅱ. TTCN 확장에 관한 연구동향

ISO/IEC JTC1/SC21의 WG1에서는 적합성 시험 표기 기법인 TTCN을 정의하여 최근에 국제 표준화 작업이 마무리되고 있다[1]. 표준화 동향의 특징을 살펴보면, 첫째, 표의 머리부분은 가로형태이며 코멘트를 포함하는 일반화된 형태로 정의하고 있다. 둘째, 제약부나 시험 그룹을 나타내는 경우, 경로를 분리하여 중복 선언을 피하도록 한다. 셋째, 시험 언어로서의 표현력을 강화하기 위해 연산자를 추가하며, ASN.1(Abstract Syntax Notation one)의 이용을 확대하고 있다. 이러한 TTCN의 표준화 동향은 앞으로 이용될 많은 응용 프로토콜 구현제품들에 대한 적합성 시험 suite을 명확히 기술할 수 있도록 지원하는 것이다.

이와 병행하여, 새로운 적합성 시험 구조 형태인 멀티파티 시험 방법과 TTCN 확장에 관한 연구가 진행 중에 있다[2][3][4]. 멀티파티 시험 방법에서는 ISDN, Transaction Processing, Routing, 그리고 MHS(Message Handling System) 등에 대한 적합성 시험 방법을 연구하고 있다. 한편, TTCN 확장에 관한 내용은 병렬성의 추가와 TTCN 연산자및 연산의 추가, TTCN 제약부의 강화 등의 연구가 진행되고 있다.

1. 멀티파티 시험 방법(Multi-Party Test Method)

X.25와 같은 프로토콜 구현제품들은 적합성 시험을 위하여 하나의 하위 시험기(Lower Tester)와 상위 시험기(Upper Tester)의 PCO를 통해 IUT(Implementation Under Test)를 제어, 관찰함으로써 적합성 여부를 판정할 수 있다. 그러나, ISDN과 같은 응용 프로토콜 구현제품들은 분산된 여러 개의 엔터티들이 여러 개의 프로토콜을 동시에 주고 받을 수 있기 때문에 단일 시험 구조로는 병렬로 수행되는 프로토콜 행위를 시험할 수 없다. 이에 대한 적합성 시험을 위해서는 병렬로 수행되는 여러 개의 시험기와 IUT의 여러 프로토콜 행위를 동시에 시험, 관찰할 수 있는 여러 개의 PCO를 갖는 멀티파티 시험 구조가 필요하다[2][3].

ISDN 프로토콜과 같은 경우에는 B와 D 채널 등 다른 채널을 통해 동시에 통신해야하기 때문에 기존의 적합성 시험 방법으로는 몇 개의 채널을 동시에 시험할 수 없다. 그러므로, 각 채널에 따라 몇 개의 하위 시험기를 구성하고, 여러 개의 PCO를 통해 병렬로 프로토콜 시험 행위를 수행하여 IUT를 시험, 관찰하는 복수 시험 구조를 구성하여 적합성 여부를 판정할 수 있도록 한다. 이러한 복수 시험 방법은 ISDN 프로토콜뿐아니라 앞으로 정보 통신 분야에서 중요하게 이용되는 트랜잭션 처리 프로토콜, 네트워크 관리 프로토콜, 멀티미디어 프로토콜등의 복수 프로토콜 구현제품의 시험에 적합한 시험 방법이라고 할 수 있다.

멀티파티 시험 구조에서 IUT의 하위 부분을 제어, 관찰하는 시험기를 병렬 하위 시험기, 상위 부분을 제어, 관찰하는 시험기를 병렬 상위 시험기라고 한다. 또한, 이들 병렬 상·하위 시험기간의 제어, 조정을 하는 주 하위 시험기와 주 상위 시험기가 있으며, 이들간에는 시험 조정 절차에 따라 프로토콜 시험 행위를 수행한다. 그림 1은 일반적인 멀티파티 시험구조를 나타낸다. 하나의 IUT는 동시에 여러 방향의 프로토콜을 다루는 부분들과 이들간의 조정을 이루는 부분으로

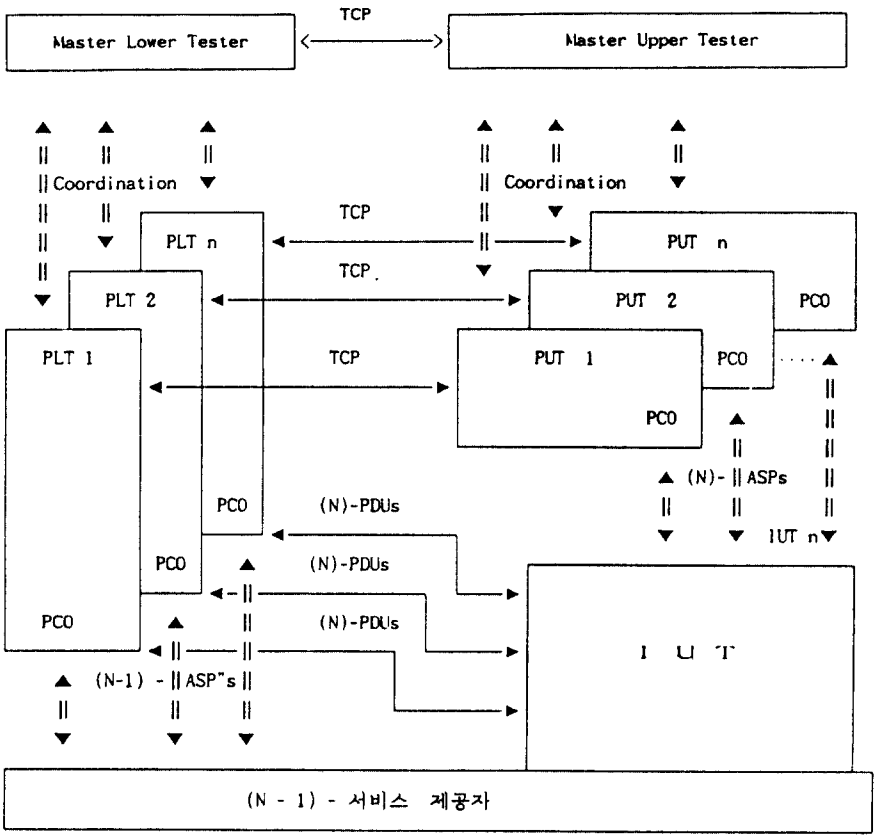


그림 1. 일반적인 멀티파티 시험 구조
Fig. 1. Generalized Multi-Party Test Architecture

나눌 수 있다. IUT는 Transaction Processing 프로토콜과 같이 각 방향에서 같은 종류의 프로토콜을 사용하는 것이 있고, ISDN 프로토콜과 같이 B와 D 채널 등의 다른 종류의 프로토콜을 사용하는 것이 있다.

2. TTCN의 병렬성 확장

OSI(Open System Interconnection) 응용 프로토콜들은 여러 엔터티들이 병렬적으로 동시에 수행하는 프로토콜 행위를 포함한다. 그러나, 기존의 TTCN은 병렬적인 프로토콜 시험 행위를 표기할 수 없으므로, TTCN을 이용하여 이러한 구현제품들의 적합성 시험 suite를 작성하기 위해서는 TTCN이 병렬성을 포함하도록 확장해야 한다. 이 절에서는 TTCN 확장시 고려해야 할 사항과 TTCN이 병렬성을 나타낼 수 있도록 새로이 정의한 병렬 시험 요소와 그들간의 상호작용에 대해 관찰한다[5][6].

2.1. TTCN 확장에 관한 요구 사항

TTCN은 적합성 시험 suite를 명세하기 위해 ISO에서 표준으로 정의한 시험 표기 기법이다. TTCN에 병렬성을 부가하여 확장하기 위해 다음의 사항을 고려하여야 한다. 첫째, 기존의 TTCN을 크게 변형시키지 말고 확장에 따른 비용과 노력을 최소화하도록 한다. 둘째, 병렬로 수행하는 시험 요소의 행위를 시험 suite에 명세해야 한다. 셋째, 시험 요소간의 상호 작용을 위한 동기화 메카니즘을 설계해야 한다. 그리하여, 전체시험 suite를 쉽고 명확하게 이해할 수 있도록 TTCN

확장시 고려해야 한다. TTCN은 형식적 명세 기법(Formal Description Technique)에 비해 비형식적인 언어 형태이기 때문에 프로토콜의 시험 행위를 알고 쉽고 명확하게 표기하기 위하여 보다 형식적인 개념을 추가하여 시험 표기 언어로 발전시켜야 한다.

2.2. 병렬 시험 요소(Parallel Test Component)

하나의 시험기는 하나이상의 시험 요소(test component)로 이루어진다. 이 때, 여러 개의 시험 요소들이 병렬로 프로토콜 시험 행위를 수행하는 경우, 이러한 시험 요소들을 '병렬 시험 요소(Parallel Test Component, PTC')라고 정의한다. 하나의 시험요소가 PCO(Points of Control and Observation)를 통해 주고 받으며 시험 행위를 수행하는 것처럼, 병렬 시험 요소간의 통신은 조정점(Coordination Point, CP)을 통하여 서로 조정 메시지(Coordination Message, CM)를 주고 받음으로써 상호작용하도록 한다.

2.2.1. 병렬 시험 요소의 선언

TTCN에서 모든 형태 선언은 시험 suite 단계에서 이루어지기 때문에 병렬 시험 요소도 다른 형태처럼 시험 suite 단계에 추가하여 선언되어야 한다. 병렬 시험 요소 선언표는 표 1과 같이 병렬 시험 요소 이름과 병렬 시험 요소 역할, 코멘트 필드로 구성되며, 시험 요소 역할에서 병렬 시험 요소는 'PTC'로 표기하고, 병렬 시험 요소 중 시험 요소간의 조정과 판정값을 종합하는 역할을 하는 주 시험 요소는 'MTC'로 표기한다. 또한, 병렬 시험 요소 이름은 시험 suite 내에서 유일한 이름을 가질 수 있도록 정한다.

표 1. 병렬 시험 요소 선언
Table 1. Parallel Test Component Declarations

Parallel Test Component Declarations		
TC Name	TC Role	Comments
TCIdentifier	MTC | PTC	FreeText

2.2.2. 병렬 시험 요소 구성

적합성 시험 방법 중에 국부 시험 방법, 분산 시험 방법, 조정 가능한 시험 방법 등은 하나의 하위 시험기와 하나의 상위 시험기, 각 시험기의 PCO, 이들 시험기간의 CP등으로 구성되며, 원격 시험 방법은 상위 시험기를 구성할 수 없으므로 하나의 하위 시험기와 그 PCO로 시험 구조를 구성한다. 이러한 시험 방법의 구성 요소들을 '시험 요소 구성 선언부(Test Component Configuration Declarations)'로 시험 suite 단계에 추가하여 정의하도록 한다. 이 시험 요소 구성은 시험 사례 동적 행위부의 머리부분에 구성 이름을 표기하므로써 시험 요소나 PCO, CP들을 이용할 수 있게 된다.

표 2. 시험 요소 구성 선언
Table 2. Test Component Configuration Declarations

Test Component Configuration Declarations			
Config Name	TCs Used	PCOs Used	CPs Used
ConfigIdentifier	TCIdentifier	PCOIdentifier	CPIdentifier

2.2.3. 병렬 시험 요소 행위 기술

하나의 시험 사례는 앞절에서 언급한 참조된 병렬 시험 요소 구성을 표기해야하며, 병렬 프로토콜 시험 행위를 포함하는 여러 시험 요소들이 병렬로 수행한다는 것을 나타내야 한다. 따라서, 시험 사례 동적 행위부의 머리부분에 참조된 병렬 시험 요소 구성의 이름과 각 시험 요소들을 병렬 시험 요소로 결합하는 필드를 새로 추가하여 병렬 시험 요소의 행위를 기술하도록 한다. 표 3에서 'PTC Configuration'과 'PTC Binding' 필드를 추가하여 시험 사례 동적 행위표를 새로 구성하였다. 이 때, 각 시험 요소의 행위 기술은 로컬 트리나 라이브러리 형태인 시험 단계 동적 행위로 나타낼 수 있으며, 병렬 시험 요소로 결합하는 경우 이를 병렬 트리(parallel tree)라고 칭한다.

표 3. 확장된 시험 사례 동적 행위
Table 3. Extension of Test Case Dynamic Behaviour

Test Case Dynamic Behaviour					
Test Case Name : TestCaseIdentifier Group : TestGroupReference Purpose : FreeText PTC Configuration : ConfigIdentifier PTCBinding : TCIdentifier : =TreeIdentifier / TestStepIdentifier Default : DefaultIdentifier Comment : FreeText					
Nr	Label	Behaviour Descrip.	Constraint Ref.	Verdict	Cmmt
:	:	:	:	:	:
:	:	:	:	:	:

2.3. 병렬 시험 요소간의 상호작용(Cooperation)

병렬로 수행되는 시험 요소들은 시험 목적을 이루기 위해 그들간의 상호 작용으로 병렬 시험 요소간의 통신과 동기화가 필요하다. 병렬 시험 요소간의 통신은 프로세스간의 통신 방법을 이용할 수 있다. 프로세스간의 통신 방법에는 공유 변수(shared variable)와 메시지 전달(message passing)의 두 가지 데이타 통신 방법을 고려해 볼 수 있다. 병렬 시험 요소들은 분산 환경에서 서로 떨어져 있어 정보를 공유할 수 없는 경우에 생기므로 메시지 전달 방법을 이용하여 서로 정보를 교환하도록 한다.

병렬 시험 요소가 메시지 전달 방법을 통해 통신하는 경우, 두 가지의 동기화 방법이 있다. 병렬 시험 요소내에 버퍼를 두는 비동기적(asynchronous) 방법과 버퍼없이 데이타를 처리하는 동기적(synchronous) 방법이 있다. 기존의 TTCN에서 비동기적 방법을 지원하며 병렬 시험 요소의 비결정적인 시험 행위를 표기하기 위해 병렬 TTCN에서는 비동기적 메시지 전달 방법을 이용하여 병렬 시험 요소간의 상호작용을 수행한다.

2.3.1. 조정점(Coordination Point)

시험 요소간의 상호 작용을 위해 '조정점(Coordination Point, CP)'이라는 상호 작용점을 두어 메시지를 주고 받음으로써 동기화와 통신을 하도록 한다. 조정점은 PCO와 유사한 형태를 지니며, 양방향으로 각각 하나씩의 큐(queue)를 두어 비동기적인 통신을 수행한다. CP의 형태는 PCO와 비슷한 방법으로 시험 suite 단계에 '조정점 형태 선언(CP Type Declarations)'을 추가하여 선언해야 한다. 그 역할 필드에서는 병렬 시험요소 이름사이에 '<->'라는 기호를 이용하여, 그 CP가 두 개의 병렬 시험 요소 사이에서 조정점 역할을 한다는 것을 나타낸다.

표 4. 조정점 형태 선언
Table 4. Coordination Point Type Declarations

CP Type Declarations			
Name	Type	Role	Comments
CPIdentifier	CPTypeidentifier	PTCIdentifier <->PTCIdentifier	FreeText

2.3.2. 조정 메시지(Coordination Message)

시험 요소간의 조정점을 통해 상호작용을 하기 위해 PDU와 비슷한 형태를 지닌 '조정 메시지(Coordination Message, CM)'라는 메시지를 교환함으로써 동기화와 통신을 수행할 수 있다. 이 조정 메시지는 병행 시험 요소 수행의 시작, 중지, 재시작등의 동기화 메시지 뿐 아니라, 타이머 값, 판정값등의 정보를 전달할 수 있다. '조정 메시지 형태 정의(CM Type Definitions)'도 선

표 5. 조정 메시지 형태 정의
Table 5. Coordination Message Type Definitions

CM Type Definitions		
CM Name : CMIdentifier CP Type : CPTypeidentifier Comments : FreeText		
Field Name	Field Type	Comments
FieldIdentifier	Type	FreeText

언부에 추가하여 나타내며, 조정 메시지 이름과 조정 메시지가 전달되는 조정점, 그리고 그 필드 정보를 나타낸다.

Ⅲ. 객체 지향 병렬 TTCN의 설계

최근에 객체 지향 이 프로그래밍 언어 뿐 아니라 데이타베이스, 운영체제, 인공지능, 하드웨어 설계, 응용 및 소프트웨어 설계 방법론 등에 확산되어 커다란 영향을 미치고 있다[7]. 또한, 분산 환경에서 복잡하고 병행으로 수행되는 프로토콜 행위를 기술하기 위해, 프로토콜 명세 기법인 LOTOS와 ESTELLE과 같은 형식적 명세 기법(FDT)도 객체 개념을 도입하여 객체 지향 FDT를 정의하고 여러 응용 프로토콜들을 명세하려는 추세에 있다.[8][9][10][11]. 따라서, 객체 지향 FDT로 명세하고 구현한 프로토콜 제품들의 적합성 시험을 하는 경우, 시험 표기 기법인 TTCN에 객체 지향 개념을 도입하여 시험 suite을 작성해야 한다.

이 장에서는 객체 지향 개념의 주요 요소인 객체, 클래스, 상속 개념을 고찰하고, 이러한 개념을 TTCN에 적용하여 추상적 데이타 형태를 새로이 정의하고, 병렬 시험 요소를 객체로 모형화하며, 시험 요소 정의에 있어서 클래스와 상속 개념을 TTCN에 맞도록 새로이 정의한다. 특히, 객체 개념이 병렬성을 지원하기 때문에 우리가 설계한 확장된 TTCN을 '객체 지향 병렬 TTCN'이라 부르기로 한다.

1. 객체 지향 개념

객체 지향 TTCN 모형을 설계하는데 있어서 우선 객체 지향 개념을 명확히 정의할 필요가 있다. 객체 지향 개념은 처음 프로그래밍 언어인 Simula에서 소개되어 각 언어나 시스템에 맞게 여러가지로 정의되어 사용되었으며, 이후 객체(object), 클래스(class), 상속(inheritance)의 요소가 기본을 이루며 실세계의 모든 엔터티들이 시스템 내에서 객체로 모형화하여, 각 개체를 인스턴스로 표현하고 공통된 인스턴스를 하나의 클래스로 구성하여 개념을 정립하고 있다[12]. 객체 지향 개념을 복잡한 시스템 설계에 적용하게 되면, 시스템 설계를 보다 빠르고 쉽게 해주며, 신뢰성을 증대시키고 소프트웨어의 재활용이 가능하다는 장점을 얻을 수 있다.

1.1 객체(object) 개념

객체는 자신의 정보, 데이타와 그 데이타를 액세스하고 처리할 수 있는 메소드(method)를 가지고 있으며, 자신의 데이타를 저장하는 메소드를 이용하여 그 객체의 데이타에 대한 연산을 정의한다. 객체는 데이타를 저장하고 그 연산을 제공하는 독립적인 단위이기 때문에 데이타 추상화(data abstraction) 개념을 효과적으로 지원할 수 있다. 하나의 객체가 다른 객체로부터 정보를 얻기를 원하거나, 다른 객체의 상태를 바꾸려할 때, 그 객체의 메소드를 호출하여 통신하게 된다. 어떤 객체의 메소드를 호출하기 위하여, 그 객체의 이름, 메소드 이름, 몇몇 매개 변수 등을 메시지로 보내게 된다. 메시지를 받은 객체는 그 메시지에 맞는 메소드를 실행시켜 그 결과를 메시지로 보냄으로써 응답하게 된다. 이 때, 메시지는 그 메소드가 어떻게 구현되는지, 객체의 내부 상태가 어떤 지를 알 필요가 없다.

각 객체는 자신의 상태와 구현 방법등의 내부적인 사항과 객체 사이를 연관시키는 외부적인 사항을 철저히 구분하여 객체간의 인터페이스를 미리 정의된 프로토콜에 의해서만 가능하게 함으로써 데이타 추상화와 캡슐화를 지원할 수 있다. 또한, 객체의 내부적인 구현 방법이 변화되어도 이와 독립된 외부적인 인터페이스는 계속 유지할 수 있으므로 소프트웨어의 유지 보수에 있어 매우 유용하다.

1.2. 클래스(class)와 상속(inheritance) 개념

ISO에서 연구하고 있는 '개방형 분산 프로세싱(open distributed processing)'에서는 클래스를 '여러 객체들의 공통된 특성의 명세'라고 정의하였다[10]. 클래스는 같은 종류의 객체들의 집합

이라 할 수 있으며, 그 객체들의 공통된 데이타 구조와 연산을 포함하게 된다. 따라서, 하나의 클래스로부터 만들어진 객체들은 모두 같은 연산을 갖게 되며 모두 동일한 행위를 하게 된다. 클래스는 여러 값들의 공통된 애트리뷰트에 따라 분류하고, 그 값을 수행할 수 있는 연산을 한정시킨다는 의미에서 데이타 형태와 유사하다고 할 수 있다. 이러한 데이타 구조와 그 데이타를 처리하는 메소드를 '클래스 원형(class template)'라고 정의한다면, 하나의 클래스는 하나의 클래스 원형을 만족하는 모든 객체들의 집합으로 나타낼 수 있다.

상속(inheritance)은 하나의 클래스에서 정의한 메소드들을 다른 클래스가 공유하도록 함으로써 소프트웨어 재사용을 가능하게 한다. 또한, 상속은 클래스 원형간의 관계로 나타내며 '클래스 원형간의 점진적인 수정(incremental modification)'이라 정의할 수 있다. 즉, 어떤 클래스 원형 t가 주어졌을 때 그 원형을 확장하거나 수정하여 새로운 클래스 원형 s를 만들 수 있으며, 이때 s는 클래스 원형 t를 '상속한다(inherit)'고 말한다. 상속은 새로운 원형을 형성하는 데 있어 기존의 정의를 재사용함으로써 데이타의 중복을 피할 수 있어 유용하게 쓰이는 개념이다.

2. 객체 지향 병렬 TTCN의 설계

TTCN은 시험 시스템의 시험 행위를 기술하기 위한 하나의 언어라 할 수 있다. TTCN이 분산된 환경에서 모든 응용 프로토콜 구현제품의 시험 suite 작성에 적용하기 위해, 병렬 시험 행위를 표기할 수 있도록 TTCN을 확장하는 연구가 활발하게 진행되고 있다. 이와 병행하여, 병렬성을 효과적으로 지원할 수 있는 객체 모형을 갖는 객체 지향 개념을 TTCN에 적용한다면, 전체 시험 suite를 명확히 기술할 수 있으며, 시험 suite 작성시에 오류를 최소화하고, 복잡한 시험 시스템을 추상적으로 간단하게 명세할 수 있다.

TTCN은 데이타 형태들이 자신의 연산을 갖지 않으며, 하나의 시험 요소가 갖는 데이타를 로컬 환경으로 구성하기 어렵기 때문에, 독립적인 연산 능력을 갖는 객체로 이루어진 객체 지향 환경을 만들기 어렵다. 그러나, TTCN 형태 정의에 TTCN 연산 및 연산자를 결합하여 나타낸 추상적 데이타 형태를 새로이 정의하고, 하나의 시험 요소를 하나의 객체로 하는 객체 모형을 설계하므로써 TTCN의 객체 지향 환경을 구성할 수 있도록 할 수 있다.

2.1. 추상적 데이타 형태(Abstact Data Type)

객체 모형에서 기본 구조 단위는 '객체(object)'이며, 객체는 추상적 데이타 형태(Abstact Data Type, ADT)에 그 기본을 두고 있다. 객체 모형에서 객체는 외부 환경에 제공되는 연산들에 의해 완전히 명세할 수 있다는 점에서 하나의 추상적 데이타 형태라고 할 수 있으며, 객체의 내부 상태(state)와 구현 방법(implementation)은 외부에서 직접 접근할 수 없게 된다.

TTCN에서 이러한 객체를 모형화하기 위하여 TTCN 데이타 형태를 추상적 데이타 형태로 나타낼 수 있도록 해야 한다. 그러나, TTCN 데이타 형태는 연산 집합을 포함하고 있지 않기 때문에 객체를 모형화하는 데에 문제가 된다. 따라서, TTCN 데이타 형태와 그 데이타 형태에 맞는 연산들을 하나의 데이타 형태로 나타내는 '추상적 데이타 형태 정의(Abstract Data Type Definition)'를 선언부에 추가하여 추상적 데이타 형태를 표기하도록 한다.

형태 이름(type name) 필드는 TTCN에 정의한 데이타 형태나 사용자 정의 형태의 이름을 나타내며, 기본 형태(base type) 필드는 그 형태 이름이 상속한 클레스 형태 이름을 나타낸다. 따라서, 형태 이름 필드에 정의된 형태는 그 형태의 기본 형태가 갖는 모든 연산을 모두 포함할 수 있다. 그리고, 연산(operations) 필드는 그 형태의 데이타를 다룰 수 있는 연산들을 나열하여 외부와의 인터페이스 할 수 있도록 한다. TTCN 연산은 데이타 형태와 마찬가지로 "+", "−"나 "<=", AND, OR과 같은 미리 정의된 연산(predefined operation)과 사용자 정의 연산(user defined operation)으로 나누어 볼 수 있다. 미리 정의된

연산은 내부적으로 나타내며, 사용자 정의 연산은 표 7과 같이 '사용자 연산 정의(User Operation Definition)'를 통해 나타낸다. 사용자 정의 연산의 'description' 부분은 C언어등의 구현 언어로 기술하고, 외부에서 직접 접근할 수 없도록 은폐되어 있으며, 머리 부분에는 매개 변수와 결과값 형태만을 외부 인터페이스로 추상적으로 나타내어 내부 상태를 유지하도록 한다.

표 6. 추상적 데이타 형태 정의 예
Table 6. Example of Abstract Data Type Definitions

Abstact Data Type Definitions			
Type Name	Base Type	Operations	Comments
Integer	–	"+"(Interger, Integer) >Integer : SUCC(Integer) >Integer	
String	–	LENGTH OF (String) >Integer	
BitString	String	BIT TO INT (BitString) >Integer INT TO BIT(Integer) >BitString	
Res Type		"&"(ResType, ResType) >ResType	
Nat	Integer	0 >Nat "+"(Nat, Nat) >Nat :	

표 7. 내부 구현 상태를 은폐한 사용자 연산 정의 예
Table 7. Example of User Operation Defintion that hides the implenetation

User Operation Definition
Operation Name : SUCC(i : INTEGER) Result Type : INTEGER Comments : The SUCCESSOR operation
Description [hide]
int SUCC(i) int i ; {return(i+1) ; /* return the successor value of i */ }

2.2. 시험 요소와 객체

분산 환경에서 여러 엔터티들이 동시에 수행할 수 있는 복잡한 프로토콜 행위를 시험하기 위하여 전체 시험 suite를 이해하기 쉽고 모듈화하여 나타내어야 한다. ISO에서는 TTCN이 병렬적인 시험 행위를 나타내도록 병렬 시험 요소를 두어 병렬 트리 형태로 그 시험 행위를 표기하고 있으며, 이와 병행하여 이러한 시험 요소를 하나의 객체로 모형화한다면, 각 시험 요소의 시험 행위들을 모듈화, 캡슐화하여 내부 구현의 일관성을 유지할 수 있으며, 외부 환경에 대한 제한된 인터페이스를 제공하므로써 추상화를 나타낼 수 있다. 또한, 객체 개념이 병렬성을 지원할 수 있기 때문에 병렬 시험 행위를 명확하고 이해하기 쉽게 표기할 수 있다[10].

하나의 시험 시스템은 시험 요소라는 객체들의 집합으로 나타낼 수 있다. 각 객체는 자체적으로 정보 처리 능력을 갖고 있으며, 국부적인 데이타를 저장하는 로컬 메모리를 갖는다. 또한, 시험 요소의 함수와 특성들은 하나의 객체로 모형화하여 표현되며, 객체의 특성을 이용하기 위하여 그 객체에게 정보를 요구하는 메시지를 보내게 된다. 메시지를 받은 객체는 패턴 매칭(pattern-matching)에 의한 메시지를 비교하고, 그 메시지가 요구하는 일련의 시험 행위를 수행하게 된다.

TTCN에서는 시험 요소와 IUT간, 여러 시험 요소간의 통신은 비동기적 메시지 전달형태로 한다. 즉, 시험 시스템에서 객체가 수행하는 일련의 시험 행위들은 비동기적으로 진행되며, 객체간의 통신 및 동기화는 메시지 전달에 의해서만 가능하다. 두 객체간에는 PCO나 CP같은 상호 작용점을 제외하고는 공유할 수 있는 데이타를 두지 않으며, 각 객체의 정보는 메시지 전달에 의해서만 간접적인 접근을 할 수 있도록 허용한다. 이와 같은 메시지 전달 통신 방법은 객체간의 제한된 인터페이스를 허용하기 때문에 시험 요소를 객체로 모형화할 수 있는 장점이 된다.

IUT를 자신의 PCO를 통해 제어, 관찰할 수 있는 하나의 시험 요소는 ASP나 PDU 같은 메

시지를 PCO를 통해 전송하고 응답을 받는 시험 행위를 통해 IUT가 원하는 프로토콜 행위를 수행하는 지를 검사한다. 또한, 여러 개의 병렬 시험 요소가 동시에 프로토콜 시험 행위를 수행하는 경우에 시험 요소간의 조정점인 CP로 연결되어 통신과 동기화를 통해 상호작용을 한다. 시험 요소가 IUT나 다른 시험 요소에게 ASP, PDU와 CM과 같은 메시지를 전송하는 경우, PCO와 CP의 출력 큐 버퍼에 저장하고 비동기적으로 계속 시험 행위를 수행하게 된다. 메시지를 받은 IUT나 다른 시험 요소는 입력 큐 버퍼의 먼저 도착한 메시지 순으로 적절한 패턴 매칭에 의해 비교하여 해당되는 행위를 수행한다. 또한, 메시지를 보낸 시험 요소나 또 다른 시험 요소에게 요구한 메시지에 대한 응답으로 응답 메시지를 보낼 수 있다.

하나의 시험 요소를 하나의 객체로 모형화되는 경우, PCO나 CP를 통해 메시지를 받는 '시험 사건(test event)'은 객체에서의 메소드(method)로 취급될 수 있다. 이 시험 사건에 일반 수신(receive)문과 기타 수신(otherwise)문, 시간종료(timeout)문 등이 있다. 수신문들은 PCO나 CP에 메시지가 도착하는 것으로 입력 사건이라고 할 수 있으며, 외부적으로 인터페이스 가능한 지점이라고 할 수 있다. 시간 종료문은 타이머 사건이라 하며 프로토콜 타이머의 종료를 의미한다. 시간 종료문은 수신문과 달리 내부적으로 타이머가 종료하는 사건이 발생하였을 때 수행 가능하게 된다. 시험 사건은 객체로 모형화하는 데 메소드와 같은 중요한 역할을 하며, 메시지 매칭 기능과 결합하여 내부적으로 어떠한 프로토콜 행위를 수행할 지를 결정하게 한다. 시험 사건이 메시지를 받아 메시지 매칭 알고리즘이 수행되면 그 결과로 참(true), 거짓(false) 값을 나타내게 되는데, 참인 경우 나머지 '시험 행위(test behaviour)'를 수행한다. 이러한 시험 행위에는 할당문(assignment-list)과 타이머 작동(timer-operation), GOTO 문, 전송(send)문 등이 있다. 각 객체에서의 시험 행위는 내부적으로 순차적으로 수행되어 판정값을 결정하게 되고, 외부에서 직접적으로 그 내용을 관찰하거나 상태 변수를 바꿀 수 없도록 한다.

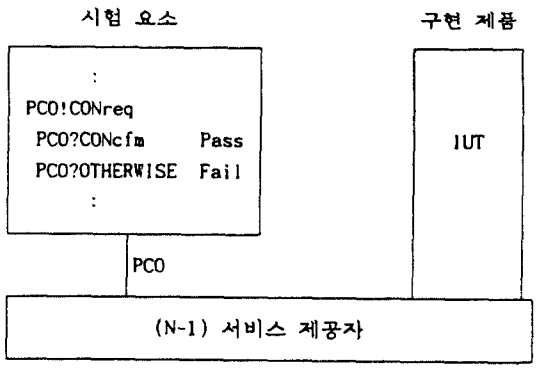


그림 2. 객체로 표현된 시험 요소 예
Fig. 2. Example of a test component represented a object

시험 요소가 객체로 모형화되는 경우, 시험 대상인 실제 프로토콜 구현제품 IUT는 암상자(black box)로 표현될 수 있다. 그림 2에서와 같이, 하나의 시험 요소가 회선 연결을 위한 connect request 메세지를 IUT로 보내는 경우에, IUT는 그 메시지에 해당하는 시험 사건을 통해 메시지 매칭이 참인 경우 내부적으로 프로토콜 행위를 수행하고, 그 결과값으로 시험 요소에게 그 메시지에 대한 응답을 보내게 된다. 시험 요소가 connect conform 메시지를 받는 경우에는 나머지 시험 행위를 수행한 후, 내부적으로 Pass라는 판정값을 할당하며, 그외의 다른 메시지인 경우에는 'OTHERWISE'문에 해당되어 Fail이라는 판정값을 할당하게 된다.

2.3. 클래스와 상속

하나의 시험 요소가 하나의 객체로 나타내어지는 경우, 다른 객체와 독립적으로 데이타를 저장하고 처리할 수 있는 능력을 갖게된다. 그러나, 여러 개의 객체들은 동일한 시험 사건을 갖게 되는 경우가 있기 때문에, 이러한 시험 사건을 공유하여 나타내게 되면, 시험 요소들이 시험 사건을 중복하여 표기하는 것을 피할 수 있으므로, 복잡

한 시험 suite 작성을 간단하게 줄일 수 있다. 이에 동일한 시험 사건들을 갖는 객체를 하나의 집합으로 하는 '시험 요소 정의(test component defintion)'라는 클래스를 정의할 수 있는데, 이는 시험 사건들을 시험 요소 동적 행위표의 머리부분에 표기하므로써 나타낼 수 있다. 표 8은 시험 사건이 포함된 시험 요소 동적표의 머리부분을 나타내었다.

표 8. 시험 사건을 갖는 시험 요소 동적 행위
Table 8. Test Component Dynamic Behaviour including test events

Test Case Dynamic Behaviour					
TC Name : TCidentifier Group : Groupidentifier Object : FreeText TC Events : Receive stmt /OTHERWISE stmt /? TIMEOUT Default : Defaultidentifier Comment : FreeText					
Nr	Label	Behaviour Descrip.	Constraint Ref.	Verdict	Cmmt
:	:	:	:	:	:

하나의 시험 요소 정의에 실제로 해당되는 값이 할당되고 내부적인 행위 기술이 표현되었을 때, 시험 요소 정의로부터 하나의 객체, 즉 시험 요소가 실체화(instantiation)된다고 말할 수 있다. 즉, 하나의 시험 요소 정의는 클래스에 해당되고, 그것으로부터 실체화된 시험 요소는 객체로 나타낼 수 있다. 동일한 종류의 시험 요소는 하나의 시험 요소 정의로 나타내어 동일한 시험 사건인 메소드들을 포함하게 된다.

시험 요소 정의라는 클래스는 객체와 같이 클래스의 데이타 구조와 그 데이타를 처리하는 메소드를 갖는 '시험 요소 원형(test component template)'이라는 클래스 원형을 정의할 수 있다. 따라서, 하나의 시험 요소 원형으로부터 정의한 데이타 구조나 메소드를 다른 시험 요소 원형이 그 특성을 계승하는 '상속(inheritance)'을 나타낼 수 있다.

객체로 나타내는 시험 요소의 프로토콜 시험 행위 기술은 로컬 트리나 시험 단계동적 행위로 나타내며, 특히 시험 단계는 라이브러리 형태로 그룹지어 나타난다. 이러한 그룹 형태는 계층적으로 구성되며, 서로 연관성을 갖게 된다. 이러한 그룹 형태를 상위 시험 요소와 하위 시험 요소로 계층적으로 구성하여 하위 시험 요소가 상위 시험 요소의 특성을 상속할 뿐 아니라 자신의 특성을 추가하여 구성하도록 한다. 따라서, 우리는 상속된 하위 시험 요소는 '부클래스(subclass)'라고 하며, 상위 시험 요소는 하위 시험 요소의 '전클래스(superclass)'라고 정의할 수 있다.

부클래스 형태로 나타난 하위 시험 요소는 추상적 데이타 형태에서의 부형태(subtype)와 같은 특성을 지닌다. 예를 들어, 데이타 형태 "자연수"를 나타내는 경우, 자신의 전클래스인 "정수"라는 데이타 형태로부터 그 특성을 상속받게 된다.

표 9. 정수의 부형태로 나타낸 자연수 데이타 형태 정의
Table 9. Nat Type Definition represented the subtype of Integer

Abstact Data Type Definitions			
Type Name	Base Type	Operations	Comments
Integer	–	"+"(Interger, Integer) →Integer : SUCC(Integer)→Integer	
Nat	Integer	0 →Nat PRED (0) →0 "+"(Nat, Nat)→Nat :	

위에서 추상적 데이타 형태에서 "정수"와 그 부형태인 "자연수"를 정의하였다. 집합의 관점에서 볼 때, "정수(Integer)⊃ 자연수(Nat)"의 관계로 표현할 수 있으며, 부형태를 나타내는 경우에는 각 경계에서의 연산을 추가하여 나타낼 수 있다. 자연수 Nat는 0을 포함하도록 하였고, 0에 대한 PRED(predecessor)를 0으로 하는 연

산을 새로이 추가하므로써 Nat에 대한 하한 경계를 정의하였다. 따라서, 자연수 Nat는 정수의 연산을 모두 이용할 수 있으며, 더우기 경계를 정의하는 자신의 연산을 새로이 추가하여 이용할 수도 있다. 또한, "+" 등의 연산을 수행하는 경우, 연산 overloading이 가능하며 이를 확장하여 polymorphism을 구현할 수 있다.

시험 요소 원형에서 추상적 데이타 형태에서와 같이 부형태를 정의하기 위해, 시험요소 행위 기술 부분에서 선택(alternative)문을 이용하여 부클래스를 정의할 수 있다. 하위 시험 요소가 상위 시험 요소의 시험 행위 특성을 상속받는 경우, PCO나 CP를 통해 메시지를 수신할 수 있는 시험 사건을 선택문으로 나타내어 진 경우, 새로운 시험 사건을 추가하여 상속을 나타낼 수 있다. 이 경우, 하위 시험 요소는 상위 시험 요소의 특성뿐 아니라 자신의 시험 사건을 새로운 메소드로 추가하므로써 부클래스로 나타낼 수 있다. 하나의 시험 요소가 또 다른 시험 요소로부터 상속되었다는 것은 시험 요소 동적 행위표의 머리 부분에 "inherit"를 추가하여 나타낸다.

표 10. 시험 요소간의 상속 관계
Table 10. Inheritance relation between two test components

Test Component Dynamic Behaviour	
TC Name : TC1 Group : TC-LIB / Inherit : Object : TC Events : PCO1?② : Default : Comment :	
Behaviour Descrip.	Verdict
PCO1?②	(Pass)

(a) 상위 시험 요소

Test Component Dynamic Behaviour	
TC Name : TC2 Group : TC-LIB / Inherit : TC1 Object : TC Events : PCO1?② : PCO2?③ ; Default : Comment :	
Behaviour Descrip.	Verdict
PCO2?③	(Pass)

(b) 하위 시험 요소

위의 예에서, 시험 요소 TC2는 TC1에 "PCO2?③"이라는 선택문을 추가하므로써 TC1의 특성을 상속하게 된다. 즉, TC2는 TC1이 처리하는 "PCO1?②"라는 시험 사건뿐 아니라 자신의 시험 사건인 "PCO2?③"를 처리할 수 있다. 이 때, TC1이 교착 상태(deadlock)가 생기면 TC2도 같은 상황에서 교착 상태가 발생해야 하는 특성을 지녀야 한다. 따라서, 새로 추가하는 시험 사건이 TC1의 교착 상태(deadlock)를 피하게 되는 특성을 갖는다면 상속의 성질을 잃게 된다. 따라서, 상속되는 시험 요소는 PCO와 CP가 다른 선택문이 추가되어야 하며, 같은 PCO나 CP를 갖는 시험 사건으로 추가하더라도 "OTHERWISE" 문과 같은 시험 사건을 추가하여 교착 상태를 피하는 특성을 지니지 않도록 고려해야 한다. 이러한 상속 개념은 시험 사건을 재활용할 수 있어, 시험 행위 기술 문장을 줄일 수 있다는 잇점이 있다.

Ⅳ. 결 론

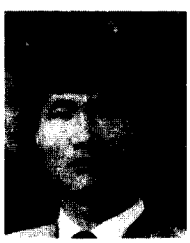
본 논문에서는 우리는 적합성 시험 표기 기법인 TTCN에 객체 지향 개념을 추가하고, 병렬성을 지원하는 객체 지향 병렬 TTCN 설계를 위한 몇 가지 내용을 제안하였다. 객체의 기본이 되는 데이타 추상화를 위해 '추상적인 데이타 형태 정의'를 선언부에 정의하였으며, 병렬 시험 요소를 객체로 나타내기 위해 PCO나 CP를 통해 메시지를 받는 시험 사건을 메소드로 정의하여 시험 요소 동적 행위에 기술하였다. 또한, '시험 요소 원형'이라는 클래스 원형을 새로 정의하여 선택(alternative)문을 통한 시험 요소간의 상속을 나타낼 수 있도록 하였다.

앞으로 TTCN에서의 객체, 클래스에 대한 개념을 좀 더 명확히 정의하고, TTCN 확장에 관한 연구 내용, 특히 병렬 TTCN에 대한 연구 내용을 계속 참조하여 객체를 통한 병렬성의 지원을 상세히 나타낼 것이다. 또한, 이러한 객체 지향 병렬 TTCN을 ISDN과 같은 실제 응용 프로토콜에

대한 시험 suite 작성에 시험 적용할 계획이다.

參 考 文 獻

1. D. Rayner, "OSI Conformace Testing", Computer Network and ISDN Systems 14, pp.79-98, 1987.
2. ISO/IEC JTC1/SC21 WG1/N 5076, "Working Draft for Multi-Party Test Methods", Aug 1990.
3. ISO/IEC DIS 9646-3, "OSI Conformance Testing Methodology and Framework Part 3 : Tree and Tabular Combined Notation".
4. Anthony Wiles, "The Tree and Tabular Combined Notationm-A Tutorial", Swedish Telecom, 1991.
5. ISO/IEC JTC1/SC21 WG1/N 5077, "Working Draft Addendum to DIS 9646-3 : TTCN Extensions", Seoul Meeting, May 1990.
6. 최 중규, 송 주석, 최 양희, "멀티파티 시험을 위한 TTCN의 확장 및 시험 적용", 정보 과학회논문지, 제 18권, 제 3호, pp.267-279, 1991년 5월.
7. 양 해승, "객체 지향과 객체 지향 언어", 정보과학회지, 제 8권, 제 5호, pp.13-20, 1990년 10월.
8. T. Mayr, "Specification of Object Oriented Systems in LOTOS", FORTE '88, North-Holland 1988.
9. E. Cusack, S. Rudkin, C. Smith, "An Object Orienteded Interpretation of LOTOS", FORTE '89, pp.265-284, December 1989.
10. S. Black, S.Gifford, "Objects and LOTOS", FORTE '89, pp.285-297, December 1989.
11. R. Sijelmassi, P. Gaudette, "An Object-Oriented Model For Estelle", FORTE '88, North-Holland 1988.
12. 윤 유식, "객체 지향 언어의 주요 개념", 정보과학회지, 제 8권, 제 5호, pp.21-28, 1990년 10월.
13. "Reflection in an Object Oriented Concurrent Language", OOPSLA '88 Proceedings, pp.306-315, Sep 1988.

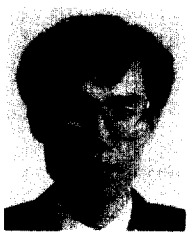


崔 重 珪(Joong Kyu CHOI) 準會員

1966년 10월 19일생

1990년 : 연세대학교 전산과학과 졸업 (이학사)

1990년~현재 : 연세대 대학원 전산과학과 석사과정



宋 周 錫(Joo Seok SONG) 正會員

1953년 3월 2일생

1976년 : 서강대학교 전기공학과 졸업 (공학사)

1979년 : 한국과학기술원 전기 및 전자공학과 졸업(공학석사)

1988년 : U.C.Berkeley 전산과학과 졸업(공학박사)

1979년~1982년 : 한국전자통신연구소 근무

1983년~1988년 : Electronic Research Lab근무

1988년~1989년 : Naval Postgraduate School 조교수

1989년~현재 : 연세대학교 전산과학과 조교수