프로토콜 개발 도구의 통합 환경

오 행 석[†] · 최 영 한[††] · 이 상 호[†††]

요 약

정보통신의 발달과 더불어 새로이 제안되는 프로토콜의 크기가 커지고 복잡해짐에 따라 프로토콜을 개발하는데 자동화 도구들이 필요로 하게 되었다. 따라서 이들 프로토콜의 개발을 지원하는 자동화도구를 개발하기 위한 연구가 활발하게 진행중이다. 그러나 현재까지 여러가지 도구들이 개발되었으나, 이들 도구들은 실험적으로 사용되고 있을 뿐 이론적 연구성과에 비해 실제 활용도가 낮은 편이다. 또한 지금까지 제안된 도구들은 대부분 개별적으로 개발되어 프로토콜을 개발하는 일련의 과정을 수행하는데 제약이 따른다. 본 논문에서는 이러한 제약을 해결키 위하여 기존에 개발된 도구들을 효과적으로 통합하여 프로토콜 개발을 위한 각종 도구들의 통합 환경을 설계하고 구축하였다. 구축된 통합환경은 프로토콜 개발 과정의 전 사이클을 제공하므로서 자동화를 통한 프로토콜 개발 기간을 단축하고 프로토콜의 신뢰성을 향상시킬 수 있다.

Integrated Environments of Protocol Development Tools

Heang Seok Oh[†] · Young Hwan Choi[††] · Sang Ho Lee[†††]

ABSTRACT

As the size and complexity of newly suggested info-communications protocols are increasing, we need automatic tools for the protocol development. There were active researches on the development of these automatic tools to support the efficient protocol developments. But, till now, even though a lot of tools were developed, those are in the experimental stage and their practicality is pretty far away regardless of significant theoretical research effort invested to produce them. Almost all the tools developed so far were usually developed individually and they show limitations in the integrated protocol developments. In this paper, to resolve these limitations, we integrated already developed tools systematically and designed and constructed an integrated environment composed of diverse tools for the effective protocol developments. The integrated environment offers a complete cycle of the protocol development processes and reduces protocol development period via the automation of using protocol development tools and improves the reliability of protocol.

1. 서 론

통신 프로토콜은 통신하는 개체간의 상호작용을 정해진 방식으로 수행하도록 하는 규율들의 집합이다. 지난 10여년 동안 컴퓨터 네트워크 및 분산시스템의 사용 증가, OSI 참조모델과 표준화된 프로토콜의 사용으로 인하여 프로토콜 공학 분야의 연구활동이 활발히 전개되었다. 프로토콜을 형식적으로 기술하는데 이용되는 FDT(Formal Description Technique)는 프로토콜 공학의 기본이 되었으며, FDT에 대한

† 정 회 원:한국전자통신연구원 선임연구원
†† 정 회 원:한국전자통신연구원 선임연구원
††† 정 회 원:충북대학교 컴퓨터과학과 교수
논문접수:1996년 9월 20일, 심사완료:1997년 4월 22일

연구 및 표준화된 FDT를 개발하기 위한 노력이 70년대 후반 이후에 ISO와 ITU를 중심으로 이루어져 이들 결과로 3가지 FDT(Estelle, LOTOS, SDL)가 국제표준으로 채택되어 널리 이용되고 있다. 이들 FDT는 추상화 표현으로 OSI 프로토콜과 서비스의 구조적 명세에 대한 표현 기능과 추상화 기능을 가지고 있다. FDT가 이용되고 그 기능이 강화됨에 따라 많은 형식적 표현방법들과 FDT를 지원하는 도구들이 개발되었다.

프로토콜 개발은 프로토콜을 공학적으로 설계하고 구현하는 과정으로서 프로토콜 설계 단계, 프로토콜 검정/검증 단계, 프로토콜 구현 단계, 프로토콜 시험 단계로 나뉘어진다[1][2].

- 프로토콜 설계 단계는 FDT를 이용하여 프로토콜 명세를 기술하고 이를 바탕으로 구문검사와 의미검사를 수행하는 과정이다.
- 프로토콜 검증 단계는 프로토콜 명세 자체의 오류들을 발견하여 프로토콜 개발 비용을 줄이고 신뢰성을 확보하기 위한 과정이다.
- 프로토콜 구현 단계는 설계 단계에서 정의된 결과물을 입력으로 하여 개발 도구를 사용하여 목적시스템에서 실행 가능한 프로토콜을 구현하는 과정이다.
- 프로토콜 시험은 구현단계에서 생성된 구현물과 주어진 명세와의 일치 여부를 판단하는 과정이다.

따라서 프로토콜 개발에 있어서, FDT의 이론적인 어려움을 극복하고 자동화를 통한 개발기간 단축 및 생산성의 향상과 조기에 오류를 제거하여 품질 향상을 도모하기 위해 여러 FDT를 지원하는 도구들이 개발되었다. 그러나 지금까지 여러 도구들이 개발되어 발표되었으나, 실험적으로 사용되고 있을 뿐 실제 활용도는 낮은 편이다. 또한 개발된 도구들의 대부분이 개별적으로 개발되어 일련의 프로토콜 개발 과정을 지원하지 못하는 바, 본 논문에서는 이들 도구들을 효과적으로 취합하여 프로토콜 개발을 위한 통합환경을 설계하고 구축하였다.

이러한 통합환경 구축을 위하여 본 논문는 2장에서는 프로토콜 개발도구 관련 현황 및 특성을 조사분석하고 3장에서는 통합환경의 구조와 기능에 대한 설명하며 4장에서는 통합환경을 이용한 프로토콜 개발의 예를 보이며 마지막으로 결론과 향후 연구과제를 기술한다.

2. 도구의 개발 현황 및 특성

FDT를 사용하여 프로토콜 개발을 지원하는 도구는 일반적으로 사용의 편리성 (사용자 인터페이스, 상업화, 도움기능, 오류처리 가능)과 확장성, 견고성 (consistency, evolution, version control, management, fault-tolerance, self-instrument 등)의 조건을 만족하여야 한다. 또한 기능적으로 명세화과정 (규격의 생성 및 관리, 규격의 문법 및 의미 체크), 검정 및 검증 과정 (규격의 문법 및 의미 체크), 구현과정 (컴파일러, 시뮬레이터) 및 시험 (능동적 및 수동적 시험)의 요구사항을 만족하여야 한다.

1987년 von Bochman의 연구에 의하면, 58개의 개발도구중 17개의 도구만이 Estelle(8), LOTOS(2), SDL(7)를 지원하였다. 따라서 1986년까지 프로토콜을 개발하는데 FDT의 이용은 아주 제한적이었고, Estelle는 구조 중심언어로 고전적인 프로그램 언어와 비슷하기 때문에 LOTOS보다 활발히 연구가 진행되었다. 1986년 이후 FDT 도구에 대한 연구는 60개 이상의 도구들이 개발될 정도로 매우 활발히 연구되었다. LOTOS는 수학적 기본 배경을 가지고 있으며 복잡한 시스템의 동작을 서술하는 강력한 표현 능력을 가지고 있어 사용자가 선호하였으나 구현하기 어려운 단점이 있다.

80년대 중반 이후에 개발된 도구들은 Chanson 의 도구 분류기준에 따라 4단계 프로토콜 개발 사이클별로 다음과 같이 분류가 가능하다[3][4][5].

- 프로토콜 설계 과정 : Book-keeping tool (프로토콜 명세의 개발, 편집, 관리, 출력 등)과 Front-end tool (프로토콜 명세의 문법과 의미 검사) 기능으로 분류됨
- 프로토콜 검증 단계 : 검정 (Validation : 프로토콜 명세의 문법적 성질 검사)과 검증 (Verification : 프로토콜 명세의 의미적 성질 검사)으로 분류됨
- 프로토콜 구현 단계 : 컴파일러, 시뮬레이터 의 기능으로 분류됨
- 프로토콜 시험 단계 : 능동적 시험 (시험 케이스의 생성과 관리, 시험 유도기)과 수동적 시험 (결과 분석기) 로 분류됨.

1995년까지 개발된 주요 도구들의 수는 〈표 1〉과 같으며, 프로토콜 설계단계 및 구현단계를 지원하는

FDT 도구들의 수가 각기 45개 및 67개로서 검증단계 및 시험단계를 지원하는 도구보다 더 활발한 연구가 진행되었음을 알 수 있다. 즉 이 부분에 대한 도구의 개발이 필요함을 알 수 있다. 〈표 2〉는 각 FDT 도구들이 지원하는 기능를 보여 준다. Estelle와 LOTOS는 한 개의 기능 이상을 지원하는 도구들이 다수 연구되었고, SDL은 두개의 기능을 통합한 도구들이 11개로 가장 많았다. 전체적으로 한 개의 기능 이상을 지원하는 도구들이 가장 많고, 4개의 기능을 통합한 도구는 Estelle에서 한 개이다. 〈표 2〉로 부터 여러가지 기능을 지원하는 통합도구를 개발하거나 자동화 도구를 효과적으로 취합하여 통합환경을 구축하는 것이 필요 함을 알 수 있다.

〈표 1〉 기능별 개발 도구 현황

FDT	설계 B	설계 F	검정, 검증 Sy	검정, 검증 Se	구현 C	구현 S	시험 A	시험 P
Estelle	5	3	3	1	10	10	2	1
LOTOS	6	7	1	3	11	11	2	3
SDL	14	11	4	1	13	6	3	0
계	45		13		67		11	

여기서 B : book-keeping tool, F : front-end tool
Sy : Syntatic, Se : Semantic
C : Compiler, S : Simulator
A : Active, P : Passive를 의미한다.

〈표 2〉 지원되는 기능 수에 따른 도구의 분류

FDT	기능의 수			
	1	2	3	4
Estelle (23)	14	8	0	1
LOTOS (33)	26	6	1	0
SDL (20)	7	11	2	0

FDT에 관련된 각종 기존의 개발 도구를 분석하고 통합하는 과정에서 향후 개발도구의 기능 및 연구방향이 다음과 같다고 사료된다.

－명세의 분석과 디자인을 지원 도구

테스트 가능성을 향상 시키고 검증을 용이하게 하기 위한 중간 객체 형태로의 변환도구들이 요구되며 생성된 중간 개체의 타당성을 증명할 수 있는 도구가 필요하다.

－검증을 지원하는 도구

현재까지는 deadlock, unspecified receipt, livelock 등을 검사하는 검정(Validtion) 중심의 연구가 진행되었으나, 의미적 특성을 검사하는 검증(Verification) 도구의 개발이 요구된다.

－프로토콜 구현을 지원하는 도구

명세를 위한 도구들과 밀접한 관계가 있으며 FDT 명세를 실행과 해석하기에 용이한 형태로 번역하는 도구들이 요구된다. 이는 입력으로서 FDT 명세를 받아 실행시킬 기계 목적 코드화가 가능한 고급언어로 기술된 소스 프로그램을 출력하는 source-to-source 번역기를 의미한다.

－테스트를 지원하는 도구

현재까지 테스트를 지원하는 도구가 부족한 편이며, 이 또한 테스트 케이스를 생성하는 방법에서 프로토콜의 제어흐름(control flow)를 기본으로 하여 테스트 케이스를 생성하였다. 향후 연구는 자료흐름(data flow)을 포함하는 테스트 케이스 생성방법에 대한 연구의 진행이 필요시된다. 그리고 데이터베이스 시스템을 이용하여 생성된 테스트 케이스를 관리하는 도구들을 포함하는 테스팅 환경을 지원하는 도구들이 개발이 요구된다.

－기타

프로토콜 명세에 정의된 시간 정보를 기본으로 하여 프로토콜 성능을 평가하는 도구의 출현이 필요하며, 또한 사용자의 이용을 편리하도록 user friendly interface 기능을 갖추어야 한다.

3. 개발도구 통합환경

본 장에서는 기존에 개발된 여러 도구의 통합시 고려하여야 할 사항과 통합환경에 활용된 자동화 도구들의 개괄적인 구조와 기능을 소개함으로써 통합환경의 역할을 제시하고 사용범위를 제안한다.

3.1 통합환경의 구축시 고려사항

본 논문에서는 현재 구축한 통합환경의 고려사항으로서 통합성, 확장성 및 수행환경을 고려하여 설계하였다.

• 통합성

현재 널리 사용되는 FDT는 Estelle, LOTOS, SDL

의 3가지로 통합환경은 이들 모든 FDT를 지원하고 있다. 여기에 포함된 도구들은 인터넷상에서 공개된 도구와 통합도구 환경을 위해 현재까지 개발된 SNUPEE (Seoul National University Protocol Engineering Environment), YACTT(Yonsei Automatic Conformance Testing Tool) 및 보유하고 있는 상업용 개발도구를 포함하고 있다[8][9][10].

• 확장성

본 통합환경은 여러개의 도구들로 구성되어 있으며, 각 도구들은 서로 독립적으로 동작되며, 기본적으로 프로토콜 개발 과정인 설계, 검정, 구현, 시험 등의 모든 개발 사이클을 지원토록하였으나 사용되는 FDT에 따라 지원되지 않는 과정이 존재하며 향후 다른 도구와의 연결가능성과 확장성을 고려하였다.

• 수행 환경

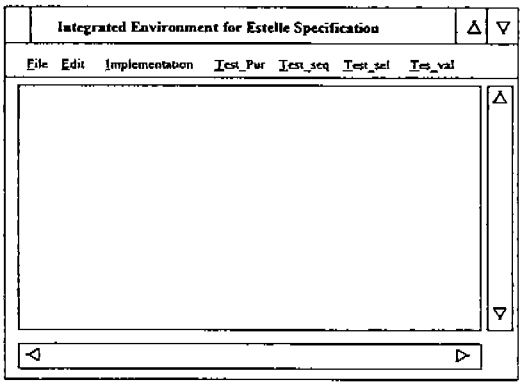
통합환경은 SunOS상에서 X/Motif를 기반으로 GUI (Graphical User Interface)를 제공한다.

3.2 통합환경의 소개

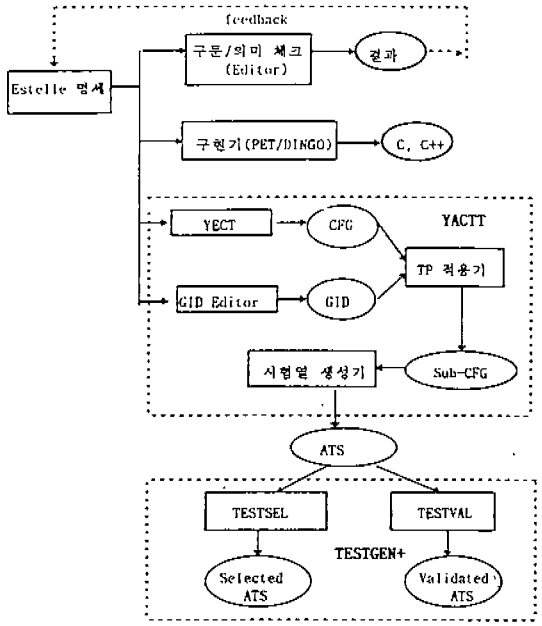
본 논문에서 제안한 통합도구는 FDT로 서술된 프로토콜 명세를 입력으로 하여 프로토콜 개발 주기를 지원하는 형태로 구성되어 있다. 여기서 FDT에 따른 프로토콜 개발 도구의 기능과 특징은 다음과 같다.

가. Estelle 개발 통합환경

Estelle 개발 통합환경은 SunOS에서 X/Motif를 기반으로 GUI(Graphical User Interface)를 제공하며



(그림 1) Estelle 통합환경의 초기화면



(그림 2) Estelle 통합환경의 기능과 구조

(그림 1)은 통합환경의 초기화면으로 명세의 편집, 구현, 시험열 생성 기능을 가지고 있다. 이들 기능을 제공하기 위해 사용한 도구 및 전체 구성은 (그림 2)와 같다.

• 프로토콜 설계 도구 (명세편집기)

프로토콜의 명세를 편집하고 저장하는 기능을 갖으며 명세 관련 도구 중에서 book-keeping tool에 해당한다.

• 프로토콜 검증 도구

현재까지 검증기능은 제공되지 않는다.

• 프로토콜 구현 도구

프로토콜 구현도구로 최종 목적 파일에 따라 Estelle −〉C^{++}의 경우는 미국 NIST(National Institute of Standards and Technology)에서 개발한 PET/DINGO를 사용하였고, Estelle −〉 C의 경우는 미국 NIST에서 개발한 NBS Prototype Compiler를 사용하였다. 먼저, PET(Portable Estelle Translator)는 FDT의 하나인 Estelle로 기술된 명세를 입력 받아 객체지향형 모델의 중간 형태(저장 가능한 객체)로 변환한다. 이는 Sun 3에서 실행되며, 5분에 Estelle로 기술된 10,000라인의 명세를 처리하며 정적모델 라이브러리(Static

Model Library)를 제공한다. 또한 DINGO(Distributed ImplementatioN GeneratOr)는 PET의 처리 후 중간 형태의 객체를 입력 받아 C++와 makefile을 생성한다. 실행 파일을 생성하기 위해 Estelle dynamic semantics을 위한 runtime library를 제공하며 X-Window library는 선택적인 사항이다. NBS 도구도 PET/DINGO와 마찬가지로 Estelle를 입력으로 하여 C 코드를 생성하는 역할을 한다.

• 프로토콜 시험 도구

프로토콜 시험 도구로 연세대에서 개발한 YACTT와 캐나다 UBC 대학에서 개발한 TESTGEN+의 일부 기능을 이용하여 통합환경을 구축하였다. 통합된 시험도구는 Estelle로 서술된 프로토콜 명세를 입력으로 하여 TTCN으로 쉽게 변환될 수 있는 추상적인 시험항목 (ATS: Abstract Test Suite) 형태로 최종 출력을 생성하는 기능을 수행한다.

먼저, 시험항목의 생성을 위해 Estelle로 서술된 프로토콜 명세를 입력으로 하여 YECT(Yonsei Estelle to CFG Translator)는 CFG(Control Flow Graph)를 만들 수 있는 transition과 GID(Global Instantaneous Description) 관련 각종 정보를 생성하고, 시험열 생성을 위한 사전작업으로 시험목적 적용기는 천이표 또는 제어흐름도에서 시험 목적에 해당하는 선택하여 트리나 그래프의 형식을 생성하며, CFG를 기본으로 해서 시험 케이스를 생성하는 시험 케이스 생성기(active testing tool)로 구성되어 있다. 여기에 추가로 GID 시퀀스 편집기(시험 목적을 GID 시퀀스로 표현할 수 있는 사용자 인터페이스 제공)와 Grapher (Transition 관련 정보를 기본으로 해서 CFG를 그림)의 부가 기능을 제공한다.

TESTGEN+은 프로토콜 시험을 위한 데이터 생성, 시험 케이스의 선택 및 시험 케이스의 검증을 위한 통합환경이다. 본 연구에서 수행하는 통합환경에서는 TESTGEN+의 기능중 시험 케이스의 선택 및 시험 케이스의 검증 도구를 활용하였다. TESTSEL은 커버리지 매트릭스에 근거하여 생성된 테스트 스위트의 크기를 줄이는데 기능을 가지고 있으며 이 두 요소는 사용자가 정의한 각종 제약 조건을 이용하여 작동되는 특성을 가지고 있다. 또한 TESTVAL은 주어진 명세서에 대한 테스트 사항을 검증하기 위하여 사용하는 도구이다.

나. LOTOS 개발 통합환경

LOTOS 개발 통합환경은 SunOS에서 X/Motif를 기반으로 GUI(Graphical User Interface)를 제공하며 (그림 3)은 통합환경의 초기화면으로 명세의 편집, 검증, 구현, 시험열 생성 기능을 가지고 있다. 이들 기능을 제공하기 위해 사용한 도구 및 전체 구성은 (그림 4)와 같다.

• 프로토콜 설계 도구 (명세편집기)

프로토콜의 명세를 편집하고 저장하는 기능을 가지며 명세 관련 도구 중에서 book-keeping tool에 해당한다.

• 프로토콜 검증 도구

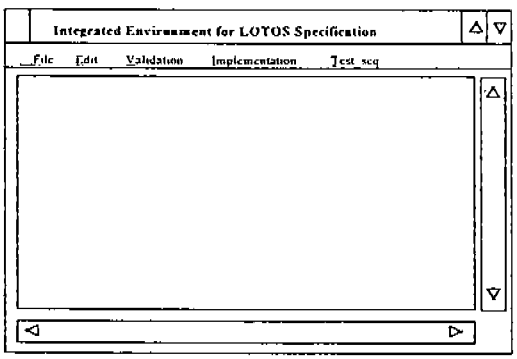
프로토콜 검증 도구로 ETSI에서 개발한 XLOLA를 사용하였으며, 이 도구는 시뮬레이션을 통한 성능 분석을 지원하고 있다.

• 프로토콜 구현 도구

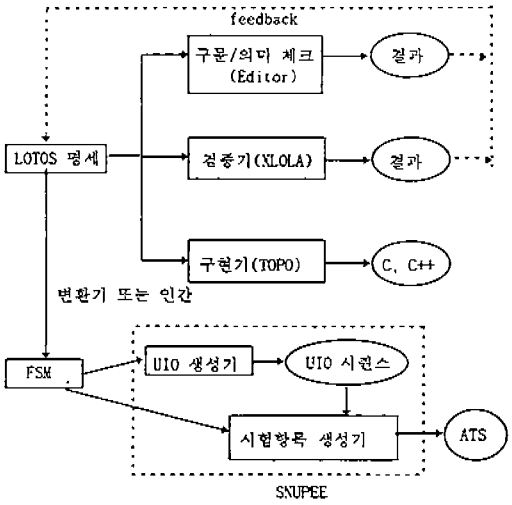
프로토콜 구현 도구로 ETSI에서 개발한 TOPO를 사용하였으며, 이 도구는 LOTOS로 기술된 명세서의 구문과 의미 체크를 하며, C 또는 Ada로 번역하는 기능을 제공한다[7]

• 프로토콜 시험 도구

프로토콜 시험 도구는 서울대에서 개발한 SNUPEE의 Test Sequence Generator를 사용하며 (그림 4)와 같이 FSM을 입력으로 하여 내부적으로 UIO 시퀀스를 생성하고 생성된 UIO 시퀀스와 FSM을 바탕으로 시험항목을 생성하게 된다[10]. 이때생성된 시험항목은 물리적인 시험항목이 아니라 논리적인 추상적인 시험항목 (ATS: Abstract Test Suite)이다. 현재는 사



(그림 3) LOTOS 통합환경 초기화면

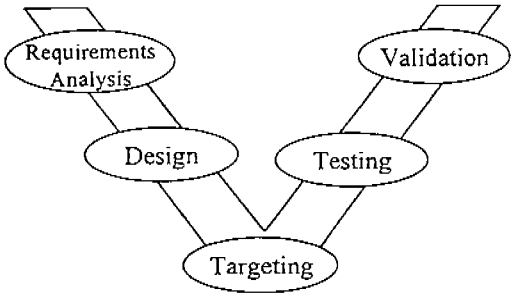


(그림 4) LOTOS 개발도구의 기능과 구조

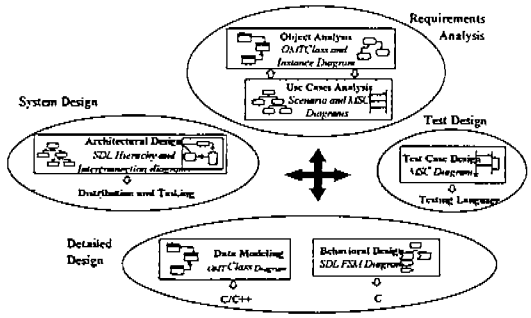
람의 개입 없이 프로토콜 명세에서 직접적인 시험항목의 생성이 이루어지지 않는다.

다. SDL 관련 지원 도구

SDL 관련 지원 도구는 현재 보유하고 있는 상업용 소프트웨어인 프랑스 Verilog사에서 개발한 Object GEODE 도구를 활용하였다. ObjectGEODE 도구의 개발 사이클과 제공하는 구조와 기능은 (그림 5)와 (그림 6)과 같으며, V-Life cycle 형태를 가지며 연속적으로 Top-down 개발 과정을 지원한다. 본 통합환경에서의 프로토콜 개발 과정에 활용은 다음과 같다.



(그림 5) ObjectGEODE 개발 사이클



(그림 6) ObjectGEODE 기능

• 프로토콜 설계 기능 (명세편집기)

사용자의 요구사항을 분석하여 실세계 환경에 적용하도록 하는 OMT(Object Modeling Technique)과 시스템의 설계 기능(SDL/GR 다이어그램을 그리기 위한 그래픽 편집과 SDL/PR 편집을 위한 텍스트 편집)을 제공한다. 또한 MSC (Message Sequence Chart)의 편집 기능도 제공하며, ITU-T Z.100(SDL-88) 및 Z.120(MSC-92)과 일치 여부와 검증과 구현을 위한 문법적인 체크를 수행하는 기능을 가지고 있다. 이는 명세 관련 도구 중에서 book-keeping tool과 front-end tool에 해당하는 편집 기능을 지원한다.

• 프로토콜 검증 기능

시뮬레이션을 통하여 프로토콜의 오류를 발견하고 시뮬레이션의 결과로 State Dump File(state의 정보)와 Edge Dump File(Test case 정보 포함하고 있으며 Test Sequence 생성에 사용)을 출력한다. 시뮬레이션 방법은 대화식과 선택된 경로에 대한 자동 시뮬레이션 기능을 제공한다.

• 프로토콜 구현 기능

Platform에 따라 C-코드를 생성하며 Debugging 기능이 내장되어 있으나, SDL 시스템에 국한되며 User Interface 기능은 제외되어 있다.

• 프로토콜 시험 케이스 생성 기능

시험 케이스를 TTCN.MP 형태로 생성하여, Test Objectives를 MSC로 기술한다.

4. Inres Protocol 적용 예

4.1 Inres 프로토콜 개요

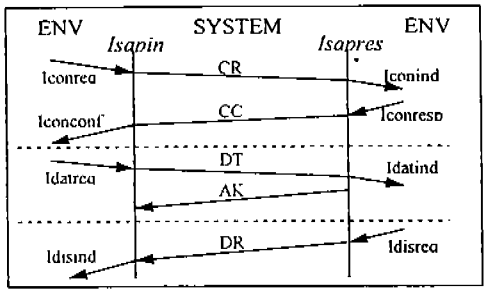
Inres 프로토콜은 connection-orient된 비동기 통신 프로토콜이다. 개시자(initiator)는 connection을 설정하고 데이터를 전송한다. 반면 응답자(responder)는 connection을 설정하거나 해제하고 데이터를 수신한다. Inres는 실제의 시스템은 아니지만 시험목적으로는 적당한 프로토콜이다. 왜냐하면 Inres는 크지 않아 이해하기 쉽고 connection, disconnection, sequence number, acknologement, 그리고 time-out/에 의한 재전송 등 기본적인 OSI 개념이 포함되어 있기 때문이다[12].

두개의 entity들간에 데이터교환이 수행되기 위해서는 4개의 서비스 프리미티브를 교환함으로써 connection이 설정되어야 한다.

- ICONreq (Inres connection request)
- ICONind (Inres connection indication)
- ICONresp (Inres connection response)
- ICONconf (Inres connection confirm)

일단 connection이 성공적으로 설정되면 개시자는 IDATreq(Inres data request)서비스 프리미티브를 통해 데이터를 전송할 수 있고, IDATind(Inres data indication) 서비스 프리미티브에 의해 응답자에 전달된다. 데이터 전송이 종료된 후에는 응답자에 의해 connection이 해제되는데 응답자가 요청한 IDISreq(Inres disconnection request)가 IDISind(Inres disconnection indication)으로 개시자에 전달된다.

Inres 프로토콜의 기본적인 프로토콜 시나리오는 (그림 7)과 같다.

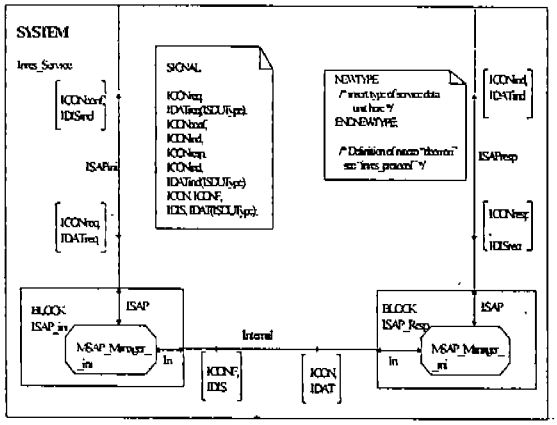


(그림 7) Inres 프로토콜 시나리오

4.2 Inres 프로토콜의 적용 예

가. 프로토콜 설계

(그림 8)은 Inres 프로토콜을 SDL 형태로 기술한 예로 Service-Provider Block인 Inres-service는 신호경로에 의해 연결된 두개의 프로세서로 구성되어 있다.



(그림 8) Inres 프로토콜의 SDL/GR 형태

나. 프로토콜 구현 및 검증

Geode 도구를 이용하여 시뮬레이션 후 Edge Dump file을 생성하기 위하여 Inres프로토콜에 대해 다음과 같은 두개의 가정을 한다.

- TIMER는 없다고 가정한다.
- 데이터의 손실이 없다고 가정한다.

GEODE simulator를 이용하여 얻어진 Edge Dump File은 〈표 3〉과 같은 상태천이도와 프로세서 블럭에 대한 Prototype의 프로그램이 얻어진다. 여기서 프로토콜 시험케이스 생성에 사용되는 상태천이도는 다음과 같은 정보로 구성된다[11].

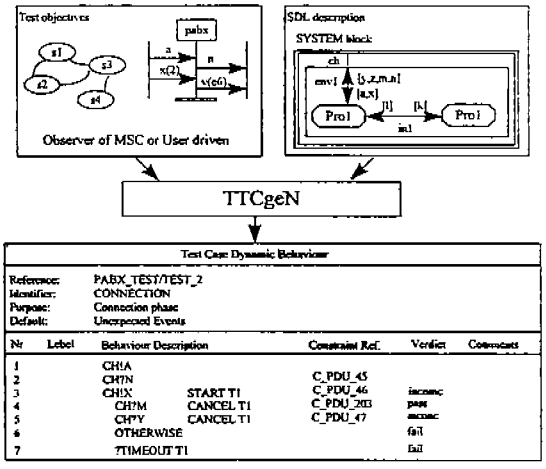
- 전체 상태의 수, 전체 edge의 수
- 각각의 edge에 대한 정보
- starting 상태의 수, ending 상태의 수
- 관련된 프로세스의 identification
- 입력 서비스 프리미티브 (input PDU)
- starting/ending 프로세스의 identification
- 출력 서비스 프리미티브 (output PDU)

다. 시험열 생성

(그림 9)는 TTCgeN(TTCN Test Case Generator)를 이용하여 MSC로 기술된 Test Objective와 SDL로 기

〈표 3〉 Inres 프로토콜에 대한 상태 천이 데이터

```
175     267
$1 2
trans initiator(1):from_dis_connected_input_iconreq with iconreq
input iconreq from env to initiator(1)
output cr from initor(1) via msap to msap_manager1 (1)
$2 3
trans msap_manager1(1):from_idle_input_cr
input cr from initiator(1) to msap_manager1 (1)
output cr from msap_manager1(1) via msap2_medium_resp to
  responder (1)
$3 4
trans responder(1):from_dis_connected_input_cr
input cr from msap_manager1(1) to responder (1)
output iconind from responder(1) vis isap to env
.....
```



(그림 9) 시험 케이스 생성 과정

술된 시스템 블럭으로 부터 TTCN.MP 형태의 시험 케이스를 생성하는 과정을 보여주고 있다.

5. 결 론

프로토콜 개발을 위한 FDT들은 ISO나 ITU 등의 표준화기관에서 연구개발되고 있으며, 프로토콜을 개발하는데 FDT의 중요성이 인식되고 있다. 이는 지난 몇년 동안 이 분야에서의 많은 연구활동에 의해 증명되었고, 많은 FDT 도구들이 개발되어 사용 가능하게 됨에 따라 실제 문제에 대한 FDT의 응용이 일반화될 것이다.

본 논문에서는 프로토콜 개발을 위해 연구된 FDT 도구 역활과 기능을 소개하고 프로토콜 개발을 위한 통합환경을 설계하고 구축하였다. 특히 여기서는 개발환경이 다른 도구와의 특성과 확장성을 고려하여 통합환경을 설계하였다.

본 논문에서 제안한 통합도구는 FDT로 서술된 프로토콜 명세를 입력으로 하여 각 FDT에 따라 프로토콜 개발 주기를 지원하는 형태로 구성되어 있다. 각 FDT에 따른 통합환경은 크게 네가지 부분으로 구성되어 있다. 첫째 프로토콜의 명세와 편집 기능, 둘째는 명세의 오류를 발견하는 검증과정, 셋째는 프로토콜 구현기능, 마지막으로 넷째는 시험케이스를 생성하는 분분으로 구성되어 있다.

참 고 문 헌

[1] D.P.Sidhu, A.Chung, C.S.Chang, "Probabilistic Testing of OSI Protocols," IEEE Tran. on Comm, Vol 42, NO.7, 2432-2440, 1994.

[2] G.V. Bochman, "Usage of protocol development tools:the results of a surey," In Harry Rudian and Coliu H. West, editors, Protocol Specification, Testing and Verification, VII, IFIP, North Holland, May. 1987.

[3]. B.Sarikaya, G.V.Bochmann, E. Cerny, "A Test Design Methodology for Protocol Testing," IEEE Tran. on SE, Vol.SE-13, NO.5, pp518-531, 1987.

[4]. W.Y.L.Chan, S.T.Vuong, M.R.Ito, "On Test Sequence Generation for protocol," Symposium on Protocol Specification, Testing and Verification, p119-130, 1990.

[5]. R.E.Miller, S.Paul, "Generating Conformance Test Sequences for Combined Control and Data Flow of Communication Protocols," Symposium on Protocol Specification, Testing and Verification, 1992.

[6] W.Chun, P.D.Amer, "Test Case Generation for Protocols Specified in Estelle," FORTE'90, 1990.

[7] A.Azcorra, J.Quemada & J.Manas, "LOTOS Tool Survey," R/1.23.-UPM/1, DIT-UPM, Spain, Sept. 1992.

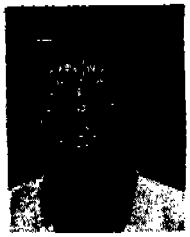
[8] 이상호, S.T.Vuong, "프로토콜 적합성 시험을 위한 통합 환경," 정보과학회 논문지, 21권 5호, pp 944-960, 1994.

[9] 장민석외 5명, "Estelle명세에서 시험경우를 자동 생성하는 통합도구," 정보과학회 논문지 (C), 2권 1호, pp 77-92, 1996.3.

[10] 김재철, 최양희, "프로토콜시험을 위한 통합환경의 설계와 구현," 정보과학회 논문지 (A), 22권 12호, pp 1695-1704, 1995.12.

[11] 이부호, 진병문, "SDL로부터의 시험시퀀스 생성에 관한 연구," 한국정보과학회 충청지부 추계학술대회논문발표집, 제7권 제1호, pp19-23, 1995. 11.

[12] D.Hogrefe, "OSI Formal Specificationcase Study :the Inres Protocol and Service," IAM-91-012, Bern University, 1992.



오 행 석

1981년 2월:한양대학교 공과대학 졸업 공학사(전자재료)

1983년 2월:한양대학교 대학원 졸업 공학석사(전자재료)

1997년 2월:충북대학교 대학원 졸업 이학박사(전자계산학)

1983년~현재:한국전자통신연구원 선임연구원(프로토콜기술연구실)

관심분야:프로토콜공학, 컴퓨터네트워크, 데이터통신

최 영 한

1981년 2월:경북대학교 공과대학 졸업 공학사(전자공학)

1992년 2월:충남대학교 대학원 졸업 이학석사(전자계산학)

1997년 2월~현재:충남대학교 대학원 박사과정 재학중(전자계산학)

1993년 2월~현재:ITU-T SG7 Q.17 Editor

1982년~현재:한국전자통신연구원 선임연구원 (프로토콜기술연구실)

관심분야:프로토콜시험, 초고속정보통신, 컴퓨터네트워크, 정보통신 표준화

이 상 호

1976년 2월:숭실대학교 공과대학 졸업 공학사 (전자계산학)

1981년 2월:숭실대학교 대학원 졸업 공학석사 (전자계산학)

1989년 2월:숭실대학교 대학원 졸업 공학박사 (전자계산학)

1992년 9월~1993년 8월:캐나다 UBC Post Doc.

1981년~현재:충북대학교 컴퓨터과학과 교수

관심분야:프로토콜공학, 시뮬레이션, 소프트웨어공학 등