論文 95-6-6-14

Design of a Test Suite Validator for Protocol Conformance Testing

Sung Hee Woo*, Byeong Ho Oh*, Sang Ho Lee* Regular Members

프로토콜 적합성 시험을 위한 테스트 스위트 검증 도구의 설계

正會員 禹 星 喜*, 吳 昞 浩*, 李 相 鎬*

이 논문은 1994년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음

ABSTRACT

Although a great deal of research activities in protocol testing have focused on formal methods for protocol test suite generation in the recent years, in practice many test suites are still developed in a heuristic manner. This gives rise to a strong need for test suite validation before they can be used in the conformance testing of implementations under test(IUT). In this paper we discuss the basic idea of validation and present an overview of TESTVAL, a tool for test suite validation based on Estelle specifications. The paper also demonstrates the validity of TESTVAL by applying it to the LAPB standard test suite. From the experiments performed we find TESTVAL a useful, practical tool for test suite validation. Some redundancies in the LAPB standard test suite were uncovered.

要 約

프로토콜 시험 분야의 많은 연구는 주로 테스트 생성에 관하여 이루어져 왔으며, 실제로 많은 테스트 스위트들이 아직도 비체계적인 방법으로 개발되고 있는 실정이다. 따라서 IUT의 적합성 시험 이전에 테스트 스위트의 검증이 매우 필요하다. 본 논문에서는 테스트 스위트 검증의 기본 개념에 대하여 정의하고, Estelle 명세서를 기반으로하는 테스트 스위트 검증 도구를 설계한다. 또한 구현된 검증 도구를 LAPB 표준 테스트 스위트에 적용하여 그 정당성을 보인다. 실험 결과 LAPB 표준 테스트 스위트에 몇개의 중복이 있슴을 발견하였다.

*충북대학교 전자계산학과
論文番號 : 95087-0227
接受日字 : 1995年 2月 27日

I. Introduction

Conformance testing of communication protocols is necessary to ensure that a protocol implementation meets its design specifications and operates correctly in communication networks[1]. Valid test suites are essential for the conformance testing of protocol implementations. However, as in the case of software testing, much research effort has focused on test suite generation rather than on validation or trace analysis[4].

In addition to the formal description techniques such as Estelle, SDL, and LOTOS, the international standard organizations CCITT and ISO have jointly developed a standard notation to describe test suites for OSI conformance testing, known as the Tree and Tabular Combined Notation (TTCN)[2]. Two TTCN test suites have been defined by ISO, one for the X.25 LAPB protocol and the other for the X.25 packet layer protocol. However, there may be potential errors with any test suite developed heuristically[5]. The presence of such errors means that we can only have limited confidence in the results of conformance testing. During conformance testing some failures may happen which can make it difficult to decide if and where a real error occurs. It is thus useful to develop an effective method for producing reliable test suites. Test validation is a way to ensure the test suites developed are valid according to some formal specification and/or will meet the testing purpose. Our approach is concerned with the first goal. Thus, the validation of a test suite is a statement defining a relationship between the test suite and the formal description of the protocol specification. The following are fundamental issues in test validation[6].

○ Are all behaviors rendering fail verdicts really invalid with respect to the formal specification of the protocol?

○ Are all behaviors rendering pass verdicts really valid with respect to the formal specification of the protocol?

The problem of test validation is basically equivalent to the one of trace analysis where a test suite can simply be viewed as a trace. Some works on the trace analysis based on LOTOS was reported in[1] and a method for trace analysis based on Estelle using Prolog was proposed in[9]. Recently a trace analyzer based on Estelle was implemented[4]. However there does not yet exist an implementation of an effective test suite validator based on Estelle.

In this paper, a test suite validator, called TESTVAL is designed and implemented and also the test validation experiments performed on the LAPB test suites translated from the LAPB standard TTCN test suite are presented. TESTVAL based on protocol specification in Estelle and ASN.1 is realized using a two-phase scheme: the first phase produces an internal representation from a given protocol specification using the frontend parser of TESTGEN, a tool for test suite generation[11] followed by the second phase which performs the validation using a simulation and symbolic evaluation method.

The remainder of this paper is organized as followed. Section 2 provides an overview of TESTVAL. Then the LAPB specification and the test validation process are presented in Section 3. In section 4 we present and discuss the results obtained from the validation of LAPB standard test suite using TESTVAL. Finally, some concluding remarks and suggestions for further work are offered in Section 5.

II. Overview of TESTVAL

For test validation, we modified and extended an existing trace analysis tool developed at the University of British Columbia[10]. The resulting test suite validator allows the checking whether a given test suite is valid with respect to a given Estelle specification. Actually the specification, which must be in P Estelle.Y, a subset of Estelle, and ASN.1[5] is first supplied to TESTVAL; subsequently we can input the test suites to be validated. The TESTGEN frontend components include an Estelle.Y parser and an ASN.1 parser, which translate the specification in Estelle.Y and ASN.1 into an internal format known as the Protocol Data Structure(PDS)[10]. Once the PDS is produced, TESTVAL performs the test suite validation in two steps : the preprocessing step and the main validation step, which operate directly on the PDS.

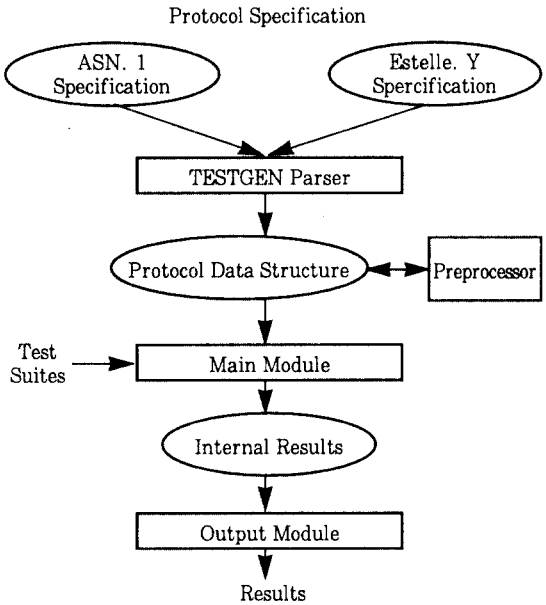

Figure 1. Structure of TESTVAL
그림1. TESTVAL의 구조

2.1 Functional Structure

As shown in Figure 1, TESTVAL consists of four major functional modules written in C language. First, TESTVAL produces the PDS by using the TESTGEN parser which was designed for test suite generations, taking into account both the control and data flows of formal protocol specifications. The preprocessor was developed for the fast and efficient processing of test suite validation. This module inserts some special attributes into the PDS such as the initial values and the types of the input service primitive(ISP)'s parameters.

The main module was designed to deal with the actual validation of the given test suites. This module starts in the initial state for each test suite validation and can currently handle two types of parameter values : integer and boolean. It validates test suites with respect to the PDS of given Estelle specifications. These data structures contain all essential protocol information, including the number of transitions from each current state and the next state of each transition. Also the current pointers of the test suites being processed are maintained in order to keep track of the current processing position in the input test suites. The output module produces the result of the test validation with some helpful information with respect to the debugging of the test suites.

2.2 TESTGEN parser

The TESTGEN parser is designed for test suite generations, taking into account both the control and the data flows of formal protocol specifications. It parses the Estelle.Y/ASN.1 specifications and translates them into the PDS. The ASN.1 parser[7] is used to parse the ASN.1 specification of

PDUs and service primitives(SP) into ASN.1 type trees. The generated ASN.1 type trees are linked to the PDS by the TESTGEN parser when the Estelle.Y specification is being parsed. As a result, the PDS includes both Estelle.Y and ASN.1 protocol information generated from an Estelle.Y/ASN.1 specification. Estelle.Y uses an Extended Transition System(ETS) to model the observable behaviors of a protocol, and ASN.1 is used to model the data representations(PDUs and SPs) of the protocol. The main differences between Estelle.Y and the normal form specification(NFS)[8] which is used in many test suite generation methods based on Estelle, are that Estelle.Y uses ASN.1 and supports more Pascal statements such as the conditional and loop statements as well as some timer primitives in the ASN.1 (TTCN) style. The structures of Input Service Primitives (ISPs), Output Service Primitives(OSPs) and Protocol Data Units (PDUs) in Estelle.Y are required to be specified in ASN.1.

The ASN.1 parser was developed using the UNIX Lex/Yacc tools, and produces an ASN.1 type tree from an ASN.1 specification. The PDS is designed to be a machine accessible form of the ETS/ASN.1-based formal description. It holds information of both the control and the data flows of a protocol specification. The ISP, OSP and PDU data types are stored in the data structures of the service primitives in ASN.1 type trees.

2.3 Preprocessor

The preprocessor consists of two submodules and converts certain types of transitions in the specification into a more suitable form for the main processing. Its function is to improve the performance and efficiency of the main module.

It produces a list containing ISP (or OSP) parameters and their types using an ASN.1 type tree generated by the ASN.1 parser. The ISP (or OSP) parameters in the list are assigned values according to the Estelle specification. It also translates the EFSM form of an Estelle.Y specification into a simple FSM form[4]. Instead of enumerating all possible combinations of service primitives and their parameters, some parts of the enumeration are realized based on the cost-effectiveness consideration. The "PROVIDED" clause of a transition corresponds to the input condition of the transition. If the clause consists of ISP parameters with values and operators such as "EQUAL" and "AND", it is considered to be a list of input symbols of a transition of the FSM. In order to make output symbols of a transition of the FSM correspond to the ones of the EFSM, the assignment statements are checked to select those consisting of OSP parameters with values.

2.4 Main Module

The main module validates given test suites with respect to the PDS of the Estelle specification. The information specified in the protocol specification should be effectively accessible to the mail module. The contents such as state, transition, constant/variable, ISP and OSP in the PDS is organized as an internal representation of the protocol state machine graph[5]. To achieve validation of test suites, it makes use of some data structures. "Paths" contains information such as the number of transitions, "from_state" and "to_state" for each transition in the path. "From_states" and "to_states" contain the state information of the system, via the values of related variables and the information on all candidate transitions at these states.

The pointer "no_proc" keeps track of the processing position of the event in the test suite. For each candidate transition, the values of variables and service primitives from the test suite are substituted to the symbolic representation of the PDS. If the predicate and the output primitives of a transition are satisfied in the test suite, then the test suite is valid according to the formal specification. The algorithm in the main module is given as follows:

Main Module Algorithm

```
int no_proc;
PDSTATE from_state;
PDTRANS transition[MAX+1];
PDSTATE to_state[MAX+1];

begin
 point first test suite;
 while not(eof)
begin
  read one test suite into buffer;
  initialize  Paths'' with from_state as initial state;
  set variables at from_state using PDS;
  find a set of executable transitions at initial state;
  while(if there is executable transition)
   begin
    set "no_proc" to 1;
    while(if there is unprocessed path)
     begin
      while(if there is unprocessed transition at from_state)
       begin
        call FIRE(the transition with from_state and to_state);
        if firable,
          add 1 to "no_proc";
        endif;
       end
      endwhile
     end
    endwhile
   endwhile
  copy the current "Paths" to new "Paths";
  if no transitions in the new "Paths"
   if the "no_proc" is less than the size of test suite,
      print "test suite is invalid";
   else
      print "test suite is valid";
    endif
   endif
   end
  point to next test suite;
  end
 endwhile
end.

Function FIRE(from_state, transition, to_state, no_proc)
begin
  assign values of parameters to the transition;
  set to_state by target state;
  copy values of variables at the from_state to the to_state;

  if the "PROVIDED" clause is false,
    return(false);
  endif;
  f the values of the variables are the same as the OSP parameters,
    save the values to to_state;
  else
    return(false);
  endif

  find a set of transitions given by PDS at
```

```
  to_state excluding transitions
    not satisfying ISP, OSPs, and their para-
    meters with respect to an event
    corresponding to no_proc and store those
    transitions in "Paths";

  if the transition is not spontaneous,
    return(true);
  endif
end
```

Each test suite consists of a sequence of events. The following is a syntactic form of one event:

"input1(or input2)/output1/output2"

where input1 and output1 are of lower interaction points of the IUT whereas input2 and output2 pertain to upper interaction points. Note that an input event from either the lower or upper layer can cause two outputs, one to the lower and the other to the upper interaction point. Each service primitive and PDU has its own name(kind of action) and parameter values list suitable for the specific action.

2.5 Output Module

TESTVAL logs the traces of states and transitions which satisfy the given test suites in a file. If a test suite is valid according to the given formal specification, TESTVAL generates a message "test suite valid" displayed on the terminal and in the log file. Otherwise, if a test suite is not valid, TESTVAL generates a message "test suite invalid" displayed on the terminal as well as storing in a log file the information regarding the position of the event in error. Using the information logged in the file and messages on the terminal, possible errors in the test suite with respect to the formal specification may be located. Whether the test suite conforms to the formal specification or not. TESTVAL keeps on processing until there is no test suite left. The logged information shows which transitions have been executed. In some cases, more than one path of the transitions may satisfy the test suite as a result of nondeterminism. In that case, we can follow those paths using "transition key"s given in the file. Every transition has a unique "transition key" identifying its location in the source of the Estelle specification. Therefore these logged informations could be helpful to debug where the error occured and what was the situation at that time.

III. LAPB specification and Test Validation

In this section, we discuss the experiments performed on the X.25 LAPB protocol via TESTVAL. The Estelle specification of the X.25 LAPB protocol is around 700 lines. The Estelle.Y and ASN.1 specification translated from the Estelle specification are approximately 1600 and 100 lines, respectively[12]. The experiments were conducted on the SUN 4/690 workstation running under UNIX.

We now explian the operation flow of the TESTVAL by applying it to the LAPB protocol. From the given Estelle specification of LAPB test suite were generated using a test case generation method such as TESTGEN[12]. The specification and test suite of LAPB protocol are fed into TESTGEN parser. The TESTGEN parser parses the Estelle specification to the PDS. As a result, the PDS includes information of LAPB protocol. The main module of TESTVAL starts its function in the initial state with the PDS and test

suite which is fed into sequentially. Test results are displayed and logged by output module.

3.1 LAPB Specification

Compared with Estelle, Estelle.Y can support only a single module specification, at most one ISP and at most two OSPs for each transition, no procedure, no function and no state list. Besides the problem of having to deal with multiple modules, two issues arise in the translation. One is how to handle while loops containing OSPs and the other is how to process transitions containing timers. These issues were partially resolved by M.C. Kim[4]. The while loop problems could be solved by using auxiliary states and variables. It is very difficult to relate the outputs from the IUT with test suites with respect to time. For simplicity, we do not consider the timer, i.e. the "delay" clause in Estelle specification is omitted during translation into Estelle.Y specification. However, it is possible to use a loosely-tuned timer to increase the performance in processing the transitions which contain a timer since this could produce fewer candidate transitions.

The Estelle.Y LAPB protocol specification has 6 states(SEND_DM, SEND_SABM, ABM, ABMONE, WAIT_SABM, SEND_DISC), 5 ISPs(ConReq, DiscReq, ResetReq, DataReq, DataIndicat), 6 OSPs(ConInd, DiscInd, ResetInd, DataInd, AckInd, DataRequest), and 1 PDU(Junk), and 132 transitions. The original LAPB protocol specification has 5 states (SEND_DM, SEND_SABM, ABM, WAIT_SABM, SEND_DISC). The ABMONE state in the Estelle.Y specification was added to handle the while loop of the original LAPB protocol specification. The major parameters of the LAPB protocol specification are given as follows.

1. frametype : frame type of SP
   I(0), RR(1), RNR(2), REJ(3), SABM(4),
   DISC(5), UA(6), DM(7), FRMR(8), BAD(9),
   contents of parenthesis are integer value to
   the corresponding frame type.
2. address : the address of station the message comes from.
   0 and 1.
3. pf : poll/final bit.
   0 and 1.
4. ns : sequence number of the coming information frame.
   between 0 and 127.
5. nr : acknowledgement from sender if pf is equal to 1.
   between 0 and 127.
6. udata : user data transmitted.

As an example, Figure 2 shows a part of the ASN.1 description of the LAPB protocol, including the DataRequest and DataIndicat service primitives.

```
LAPB DEFINITIONS ::=
BEGIN
LapbToPhy ::= CHOICE
              { DataRequest, DataIndicat }
 DataRequest ::= SEQUENCE
                 {
                  frametype    INTEGER,
                  address      INTEGER,
                  pf           INTEGER,
                  ns           INTEGER,
                  nr           INTEGER,
                  udata        INTEGER
                 }
 DataIndicat ::= SEQUENCE
                 {
                  frametype    INTEGER,
                  address      INTEGER,
                  pf           INTEGER,
                  ns           INTEGER,
                  nr           INTEGER,
                  udata        INTEGER
                 }
END
```

Figure 2. ASN.1 Description of LAPB Protocol
그림2. LAPB 프로토콜의 ASN.1표현

3.2 Validation

The whole LAPB standard test suite specified in TTCN consists of 277 test suites which are classified into eight groups[3]. The first seven groups, called Data Link Layer Test Groups DL1 to DL7 are provided to test the interactive capability of the IUT in every phase. Each test group is further divided into three subgroups according to the definitions of the improper frame, inopportune frame and proper frame. The eighth test group(DL8) is designed to test the operational correctness of the IUT system parameters, and so it is excluded from the experiments.

DL1 consists of 37 test suites in the disconnected phase; it verifies when the IUT receives DISC and sends UA or DM frame. DL2 belongs to the link disconnected phase and checks when the IUT sends DISC. DL3 is the link set up phase; it tests when the IUT sends SABM frame from disconnected phase. DL4 is the information transfer phase; it verifies when the IUT receives SABM and sends UA frame, or the IUT sends SABM and receives UA frame. DL5 pertains to the frame reject condition phase and tests when the IUT send FRMR from the information transfer phase. DL6 is the IUT busy condition phase; it checks when the IUT sends the RNR frame from the information transfer phase. DL7 is the sent reject condition phase and it verifies when the IUT sends REJ frame from within the information transfer phase.

We use the test suites translated from the LAPB standard TTCN test suites to validate the X.25 LAPB test suite. The procedure for translating TTCN into local formalism is described in[12]. It is not always possible to translate a TTCN test suite into a test suite in our local formalism. Therefore we have dealt with only 244 test suites translatable from the 277 TTCN test suites. The following are some examples of the experiments.

○ Experiment 1

Experiment 1 corresponds to the DL1_101 test suite in the LAPB standard test suite[3]. This experiment verifies that the IUT sends a DM with F=1 in response to a DISC command with P=1 received in the disconnected state. In this case, we send a DISC and wait for a DM or UA as the preamble; we send an RR command with P=1 and wait for a DM with F=1 as the postamble. This test suite is valid with respect to the specification and produces the following log file. The number in the parenthesis is the identification number of the corresponding transition.

```
Test case :
DataIndicat DISC 0 1 - - - / DataRequest
DM 0 1 - - - / - - - - - - -
DataIndicat DISC 0 1 - - - / DataRequest
DM 0 1 - - - / - - - - - - -
DataIndicat RR 0 1 - 0 - / DataRequest DM
0 1 - - - / - - - - - - -

Log file :
SEND_DM --(2)--> SEND_DM --(2)-->
SEND_DM --(5)--> SEND_DM

Result : Valid
```

○ Experiment 2

This experiment corresponds to the DL2_101 test suite in the LAPB standard test suites. This test suite verifies that the IUT sends a UA with F=0 in response to a DISC with P=0 received in the link disconnection phase. In this test suite, the tester and the IUT send DISC commands at the same time (DISC collision) after the preambles. The first two

events are preambles and are for connection setup. The test suite is valid with respect to the specification.

Test case :
DataIndicat DISC 0 1 - - - / DataRequest DM 0 1 - - - / - - - - - - - -
DataIndicat SABM 0 1 - - - / DataRequest UA 0 1 - - - / ConInd - - - - - -
DiscReq - - - - - - / DataRequest DISC 1 1 - - - / - - - - - - -
DataIndicat DISC 0 1 - - - / DataRequest UA 0 1 - - - / - - - - - - -
DataIndicat UA 1 1 - - - / DiscInd - - - - - - / - - - - - - -

Log file :
SEND_DM --(2)--> SEND_DM --(1)--> ABM --(96)--> SEND_DISC --(123)
--> SEND_DICS --(124)--> SEND_DM

Result : Valid

IV. Results Analysis

The overall experimental results are summarized in Table 1. The percentage of valid test suites varies from group to group. The differences among the seven groups are mainly due to the incompatabilities of translated test suites with respect to the Estelle.Y specification. The first three test groups behave nearly according to the LABP protocol specification in Estelle.Y. In general, most of the test suites in the LAPB standard test suite are valid with respect to the LAPB protocol specification.

However in the remaining four groups, due to the incompatabilities with the LAPB protocol specification, some test suites could not be translated and we got lower percentages of valid test suites. From the experiments of test validation we found that some of the test suites could not be validated via TESTVAL. The analysis for these experimental observations are three-fold:

1. Limitation of the Estelle.Y specification of the LAPB protocol : There is some test suites that could not be processed. For example, in the disconnected phase, the DL1_102 is specified to verify that the IUT sends a DM frame with F=0 in response to DISC frame

Table 1. Summary of experimental results
표1. 실험결과 요약

Kinds	DL1	DL2	DL3	DL4	DL5	DL6	DL7	total
No. of Test Suites	37	38	39	46	41	40	36	266
No. of Suites Processed	36	31	29	43	40	32	33	244
No. of Valid Suites	31	29	28	33	28	20	27	196
Percentage of Valid Suites	86.1	93.5	96.5	76.7	70.0	60.2	80.2	80.4

with P=0. However the Estelle.Y specification has a function dealing with that situation with F=1 and P=1 only. Thus TESTVAL could not find any corresponding transitions in the Estelle.Y LAPB specification, and consequently reported the case as invalid.

2. Redundant test suite : From a careful study of all the "invalid" test suites reported as invalid by TESTVAL, we find a few redundant test suites which may be omitted from the LAPB standard TTCN test suite. Those are in DL4 and pertain to the information transfer phase. For example, in this phase, the IUT shall transmit an FRMR frame in response to a command frame with undefined or unimplemented control field. The C/R bit can be set to either "0" or "1"and the W bit shall be set to "1" in the FRMR information field. However, this test suite is represented as two separate test suites in the LAPB standard test suite.

3. Limitation of TESTVAL : There are two examples which are manually checked to be valid but which were reported as invalid by TESTVAL. In DL4, the DL4_116 test suite verifies the IUT in the information transfer phase can manage its sending window size. For the purpose of this test, the IUT can transmit an I frame that has a N(S) value within the sending window. Acknowledgements from the tester will rotate the sending window for the DTE. The IUT window rotation must be observed over the entire valid range of sequence numbers. The IUT shall stop the window rotation when outstanding acknowledgements are not sent from the tester. The above test suites have some attributes beyond the current capability of TESTVAL. For example, currently the PDS used in TESTVAL allows up to only 10 distinct values for each parameter of one SP.

V. Conclusions

In this paper we present an overview of the structure and the function of TESTVAL , a tool for protocol test suite validation based on Estelle specification. We also demonstrate the usefulness of TESTVAL by applying to it a real life test suite, the LAPB standard test suite, with its test suites containing both the control flow and the data flow information. We present and discuss the results of the validation experiments on the entire LAPB test suite. Some interesting results are obtained from the experiments, including some redundancy in the LAPB standard TTCN test suite and limitations in the Estelle.Y specification formalism. In general, we find TESTVAL performs well as an efficient, general, flexible, and semi-automated tool which is useful for test suite validation in conformance testing. However, some aspects of TESTVAL can be improved. For example, the capability to directly accept test suite in TTCN format is an important feature that should be provided. This would bring TESTVAL closer to the complete automation of the validation process. Some extensions on the Estelle.Y language(i.e. the features supported) would be essential to allow for more complete specifications.

참고문헌

1. G. v. Bochmann, D. Desbiens, and et al, "Test Result Analysis and Validation of Test Verdicts," Proc. of IWPTS '90, Oct. 1990.
2. ISO DS 9646-3, "OSI Conformance Testing Methodology and Framework Part 3: The Tree and Tabular Combined Notation(TTCN)," 1990.
3. ISO DIS 8882, "Information technology-Telecommunications and information exchange between systems-X.25-DTE conformance test-

ing-Part 2 : Data link layer test suite," 1991.
4. M.C. Kim, "Trace Analysis of Protocols based on Formal Concurrent Specifications," Ph.D. Thesis, Dept. of Computer Science, UBC, 1992.
5. Ying Lu, "On TESTGEN, An Environment for Protocol Test Sequence Generation, and Its Applications to the FDDI MAC Protocol," Master Thesis, Dept. of Computer Science, UBC, 1991.
6. K. Naik and B. Sarikaya, "Static Validation of TTCN Test Cases," Proc. of IWPTS '90, Oct. 1990.
7. M. Sample and G. Neufeld, "Support for ASN.1 within a Protocol Testing Environment," Proc. of FORTE' 90 Nov. 1990.
8. B. Sarikaya and G. v. Bochmann, "Obtaining Normal Form Specifications for Protocols," Computer Networks Usage : Recent Experiences, pp.601-612, Elsevie Science Publishers, 1986.
9. H. Ural and R. L. Probert, "Step-Wise Validation of Communication Protocols and Services," Computer Networks and ISDN Systems 11, pp.183-202, 1986.
10. S. T. Vuong and Sang Ho Lee, "TESTGEN+ : An Integrated ENviromment for Protocol Test Suite Generation, Selection, and Validation," Proc. of FORTE '93 Oct. 1993.
11. S.T. Vuong, Sang Ho Lee and P. Zhou, "Protocol Test Validation : Principles, Tools and Examples," Invited Paper, Proc. of CFIP '93, Apr. 1993.
12. P. Zhou, "On TESTGEN, An Environment for Protocol Test Generation and Validation," Master Thesis, Dept. of Computer Science, UBC, 1992.

禹 星 喜(Sung Hee Woo) 정회원

1990년 2월 : 청주대학교 전자계산학과 졸업(공학사)
1993년 2월 : 충북대학교 대학원 전자계산학과 졸업(이학석사)
1995년 3월 : 충북대학교 전자계산학과 박사과정수료
1993년 3월~현재 : 충북대학교 시간강사
※주관심 분야 : 프로토콜 공학, 컴퓨터 보안, 소프트웨어 공학등

吳 昞 浩(Byeong Ho Oh) 정회원

1974년 : 공주교육대학 졸업후 한국방송통신대학 경영학과(학사), 전자계산학과(학사), 청주대학교 산업경영대학원 전자계산학과(석사), 충북대학교 전자계산학과 박사과정수료
1985년~현재: 충남전문대학 전자계산과 교수
※주관심 분야: 프로토콜공학, 데이타 압축, 컴퓨터 보안등

李 相 鎬(Sang Ho Lee) 정회원

1976년 2월 : 숭실대학교 전자계산학과 졸업(공학사)
1981년 2월 : 숭실대학교 대학원 전자계산학과 졸업(공학석사)
1989년 2월 : 숭실대학교 대학원 전자계산학과 졸업(공학박사)
1976년 1월~1979년 5월: 한국전력 전자계산소 프로그래머
1981년 6월~1983년 9월 : 전자통신연구소 위촉연구원
1989년 1월~1990년 12월 : 한국정보과학회 편집위원
1990년 12월~1991년 2월 : 호주 텔레콤 연구소 방문연구원
1992년 9월~1993년 8월 : 캐나다 UBC 방문연구원
1981년~현재 : 충북대학교 컴퓨터과학과 교수
1994년~현재 : 충북대학교 전자계산소장
※주관심 분야: 프로토콜 공학, 시뮬레이션, 소프트웨어 공학등