Analysis and Testing of Application Layer Protocols with an Application to FTAM

Behçet Sarikaya, Vassilios Koukoulidis, Srinivas Eswara, and Michel Barbeau

Abstract—**An experience is presented with formal specification, analysis and testing of application layer protocols. The protocol chosen as an example is the ISO file, transfer, access, and management (FTAM) protocol due to its potential for widespread use. The specification language used was the ISO standard Estelle, which was chosen to facilitate the analysis and test sequence generation of FTAM using a tool previously developed. This tool generates control and data flow graphs of the specification and derives unparameterized test sequences for each function identified by the user. We describe formal specification of application layer protocols in Estelle and translation of ASN.1 data definitions into Estelle data types. Test design tool is used to obtain functional decomposition of the control and data flow graphs. This way unparameterized test sequences are obtained. These sequences lead to a complete test suite obtained by parameterization which must be the next step. Analysis of the control and data flow graphs leads to the derivation of several properties that most of the application layer protocols must possess. The identified properties are shown to simplify the test design process.**

I. Introduction
Protocol Specification and Testing

FORMAL specifications of protocols such as in the standard language **Estelle** [7] are important as the basis of semiautomatic verification, analysis and test generation. ISO has recently defined an application layer model [9] in which fits the actual description of FTAM. This model is shown in Fig. 1 Any protocol specification must define the interactions exchanged between two peer entities called protocol data units (PDU's). For application layer PDU's the CCITT/ISO standard language called **abstract syntax notation-1** (ASN.1) is used in all the informal standards. We establish an association between ASN.1 and Estelle in Section II.

File transfer and access management (FTAM) is a layer seven protocol for transferring, accessing, and managing data files between open computer systems. FTAM also defines a common model, for files and their attributes, called the **virtual file store** (VFS). This model permits transfer, access, and management of files between systems of different manufacturers as well as of different level of sophistication. An earlier file transfer protocol design is described in [2]. Fig. 2 depicts the architecture of the FTAM service.

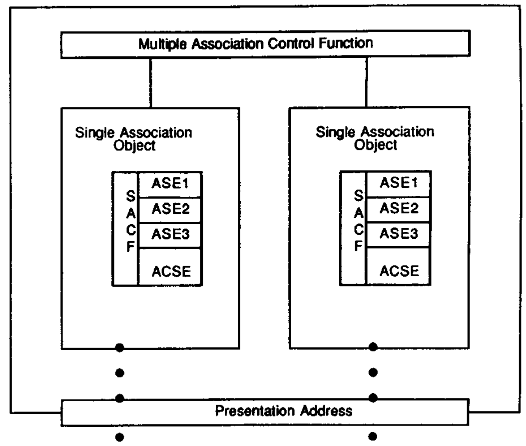


Fig. 1. Application layer model.

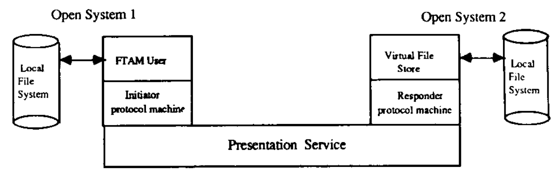


Fig. 2. Architecture of the FTAM service.

ISO has recently defined a methodology and a framework for conformance testing [8]. Parts 1 and 2 define the terminology and architectures to be used for conformance testing of the implementations under test (IUT). Part 3 defines a test specification notation called TTCN and the other parts are about test laboratory operations. Conformance testing for application layer can be done using two external test architectures depicted in Fig. 3.

II. Formal Specification of Application Layer Protocols

Formal specifications of application layer protocols/services enable automated analysis. The formal specification in Estelle of FTAM was derived from the ISO document [5] and comprises two major parts. The first part describes data types and structures used by FTAM, i.e., PDU's and ASP's. The module representing the behavior of the FTAM ASE is defined in the second part.

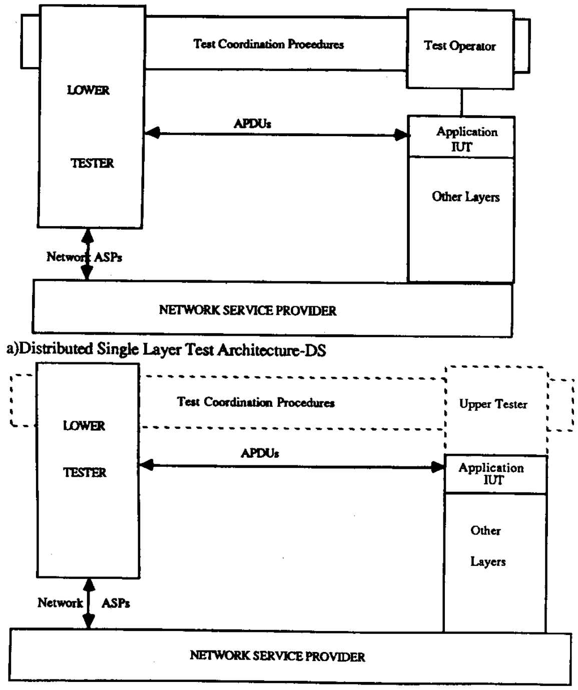
Paper approved by the Editor for Communications Protocol of the IEEE Communications Society. Manuscript received October 17, 1988; revised August 25, 1989 and May 15, 1990. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada. This paper was presented in part at INFOCOM '89, Ottawa, Ont., Canada, April 1989.

B. Sarikaya is with Department of Computer and Information Sciences, Bilkent University, Bilkent, Ankara, Turkey 06533.

V. Koukoulidis and S. Eswara are with the Department of Electrical and Computer Engineering, Concordia University, Montreal, P.Q. H3G IM8, Canada.

M. Barbeau is with the Department de Mathematics, Université de Sherbrooke, Faculté des Sciences, et d'informatique, Sherbrooke, P. Q., Canada.

IEEE Log Number 9105171.



b) Remote Single Layer Test Architecture-RS

Fig. 3. Application layer test architectures.

A. Translation of ASN.1 into Estelle Data Structures

Estelle adopts from Pascal the notation for data type and structure definitions. ISO specification of FTAM protocol uses ASN.1 [6] for defining their PDU's. Application layer service primitives are defined in a notation free manner since the implementation of a given service is a local issue. Since application service primitives correspond one-to-one to the PDU's, we assume in what follows that the ASN.1 PDU definitions can also be used for service primitives [3]. Given the need to express the FTAM application protocol in Estelle, it is necessary to translate ASN.1 definitions of PDU's and service primitives into Estelle.

ASN.1 **integer** is a simple type with positive and negative whole numbers as distinguished values. It is either a single or a (enumerated) list of values. In Estelle the distinguished values are defined as Pascal constants whereas the main type is defined of integer type. ASN.1 **bitstrings** are ordered sequences of zero or more bits. As in the case of integer type, a bitstring type can be a single or an enumerated list of values. Pascal arrays of booleans are used for representing ASN.1 bitstrings. As in the case of integer type simple bitstring type is represented as an array of boolean and the size of the array is some predetermined maximum value.

An ASN.1 **sequence** defines a new type consisting of ordered list of existing types. It structures a sequence of items whose order is significant. A sequence type is expressed in Estelle with a record type. ASN.1 **sequence of** defines, by referencing a single existing type, a new type as an ordered list of zero, one or more values of the existing type. The number of values in a sequence of is not limited. A *sequence of* is mapped to an array type item and an integer type item structured into an Estelle record. The array type item stores the sequence of values whereas the actual number of values in the sequence is assigned to the integer type item. An implementation dependant maximum sequence length is assumed. ASN.1 **set** structured type is like *sequence* type, but the items order is not relevant. A mapping similar to the one defined for *sequence of* is adopted. An ASN.1 **set of** is a structured type defined as the unordered list of zero, one or more values of an existing type. In Estelle, types of set elements are restricted to simple types and therefore can not be considered in general for mapping ASN.1 *set* or *set of* structured types. Here again we make use of a record structure similar to the one defined for *sequence of.* ASN.1 **choice** data type defines a new type from a given list of distinct types. A value can be chosen from any one of them. *Choice* structures are mapped into Pascal variant records.

```
specification FTAM_Initiator systemprocess;
{data types and structures; most of them as translated from ASN.1 using the rules given in Section
3 }
{channel definitions }
{                                        }
channel F_access_point (User, Provider );
{------FTAM Service Primitives------}
channel A_access_point(User, Provider)
{------ACSE Service Primitives------}
channel P_access_point(User, Provider)
{------Presentation Service Primitives------}

{transitions}
trans
when F.FINIrq
{F-INITIALIZErequest primitive acceptable}
provided (level=usr_corr) and (class=trans_class)
from CLOSED to INITIALIZE_PD
begin
      INIrqPDU(
      {in}
      prot_id, pres_cont_man,level,class,units,att_groups,
      rollback,contents,in_id,acc,fs_passwd,chekp_wind,PDU);
      {out}
      output A.AASSrq(0 {Version 1}, CalledAET, CallingAET,
                      ISO_8571_FTAM,PDU);
end;
{other transitions}
end FTAM_Initiator .
```

Fig. 4. Estelle specification of FTAM initiator.

B. FTAM Behavior Description

The EFSM of FTAM Initiator Entity was obtained by first translating in Estelle the transition tables that are provided in the ISO document. The result is a skeleton for the EFSM where *from/to* clauses, *when* clauses and *output* statements are defined for each transition. Specification of such behaviors as sending a PDU as a result of an incoming ASP is facilitated in Estelle by defining procedures in which parameter mapping is expressed. We consider the *Basic FTAM Initiator* protocol and specify it in two ways: a single module specification for full FTAM Initiator and a three-module specification with each module corresponding to the functional units *kernel, read,* and *write.* Fig. 4 gives an overall description of the specifications with one complete transition.

III. Functional Analysis of Application Layer Protocols

We use the tool Contest–Estl [12] to functionally analyze the FTAM specification. The resulting control and data flow

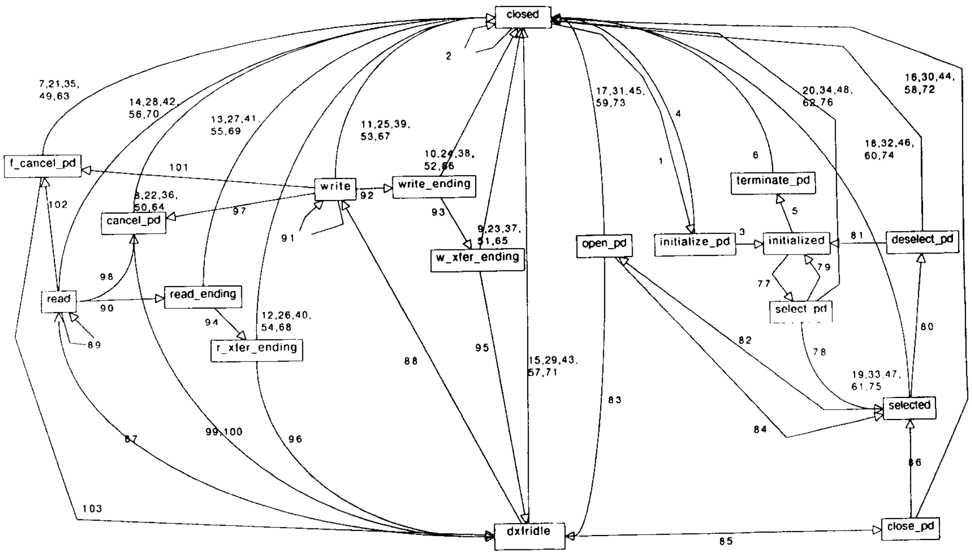


Fig. 5. FTAM initiator entity control graph.

characteristics are shown to validate the (ad-hoc) observations reported in the literature [3], [4].

Contest–Estl takes an Estelle specification and analyses it for the purpose of identifying control and data flows in the specification. Its main use is in semiautomatically producing unparameterized tests for the system described.

A possible use of the control graph is in selection of test sequences. Finite-state machine based test sequence selection techniques include transition tours, distinguishing sequences, and UIO sequences [1], [10]. The data flow graph is algorithmically partitioned into *blocks*. A block generally represents the data flow over a single variable or an input node and an output node (Fig. 6). If there are n nodes and k arcs, the complexity of the block generation process is $O(nk)$ since every node and arc are considered at most once for possible inclusion in a block.

A. Control Flow in Application Layer Protocols

Property 1) Provided clauses do not contain references to context variables.

Property 1 is the result of the fact that application layer protocols are interaction oriented, all the entity does is to respond to the interactions from the environment.

Property 2a) In a sequence of interactions, the order of service primitives and PDU's depends on PDU/ASP *result* parameters.

Property 2b) Admitted parameter values can be very complex.

Properties 1) and 2a) imply that test sequences obtained from the control graph will not contain paths that are *infeasible.*

Property 3) There is a one-to-one mapping between the set of ASP's and the set of PDU's.

Example of correspondence is F-INITIALIZE-request service primitive which corresponds to Initialize-request PDU.

Property 4) Transition tours do not contain any synchronization problems. The problem of nonsynchronizable tests does not arise in the case of FTAM because of the correspondence between ASP's and PDU's.

B. Data Flow in Application Layer Protocols

Property 1) Blocks of the data flow graph exhibit a simple structure, i.e., in general values from an I-node flow directly, without internal transformations, to an O-node. In some cases, values of the O-nodes are determined from constant D-nodes (see Fig. 6).

Property 2) After block merging every data flow graph has no incoming arcs from other functions. Specifically, block merging eliminates data flow dependencies among the data flow functions.

The control and data flow properties stated above imply the following important aspect of test generation: The control and data flow in application layer protocols can be tested independently of each other. This is very helpful since in most application layer protocols both control and data flow graphs are large and control graphs exhibit a complex structure (see Fig. 5).

IV. Functional Testing of Application Layer Protocols

A. Block Merging and Test Generation

A block Bi consists of a collection of nodes and associated arcs. In application layer data flow graphs due to the properties stated in Section III, there is no arc shaped by more than two blocks, blocks are independent of each other. We define the set SIN(Bi) (SON(Bi)) as the set of all I-nodes (O-nodes) belonging to block Bi. We also call SIL the set of labels associated with the input arcs of a node. For each block Bi,

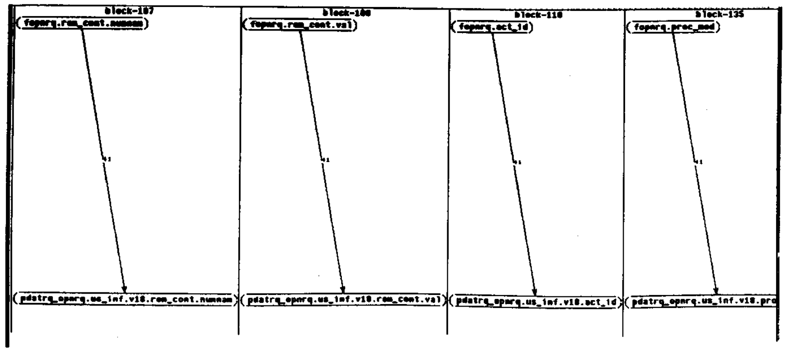


Fig. 6. An example data flow graph.

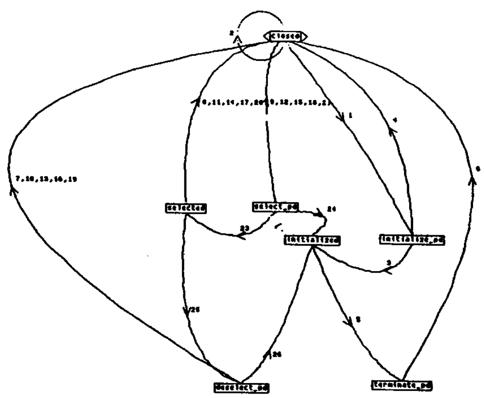


Fig. 7. FTAM initiator kernel unit control graph.

SIL(Bi) is the set obtained by the union of the SIL of the block's nodes.

Rule 1: If SON(Bi) and SON(Bj) contain parameters of the same data type then Bi and Bj are merged.

Rule 2: If SON(Bi) and SON(Bj) both contain parameters related to the same data flow function and SIL(Bj) $\supseteq$ SIL(Bi) holds then Bi and Bj are merged.

If there are initially m blocks, block merging takes no more than $m - 1$ steps since at every step the number of blocks is decreased by 1.

B. FTAM Test Design

FTAM initiator entity control graph contains 18 states and is given in Fig. 5. The *kernel* graph is given in Fig. 7. Table I lists the number of transitions in the original and normalized specifications.

Full FTAM Initiator DFG contains 261 blocks while the kernel, read and write unit DFG's contain 112, 141 and 141 blocks, respectively. We applied the rules of Section IV-A to the data flow graph of full FTAM initiator, and the functional units of kernel, read and write. Statistics about the number of obtained data flow functions are in Table II.

Next, subtours are derived from the control graphs. A subtour is a sequence of interactions that starts and ends in the initial state. The lengths of the subtours are also given in Table II. Two of the kernel unit subtours are listed in Table III. They can be used to test the *type of service* function.

TABLE I
CONTROL GRAPH STATISTICS

	# of transitions	# of normalized trans.	# of states	transition tour length
FTAM				
Initiator	36	103	18	596
Kernel	16	26	7	92
Read	20	56	10	235
Write	20	56	10	235

TABLE II
DATA FLOW STATISTICS

	# of Data Flow Functions	Length of Unparameterized Test Sequences
FTAM Initiator	21	2616
Kernel	14	419
Read	12	1089
Write	13	1257

TABLE III
SUBTOUR EXAMPLES

State	Input	Output	Transition
closed	FINIrq	AASSrq_INIrq	1
initialize_pd	AASScf_INIrp	FINIcf	3
initialized	FTERrq	ARELrq_TERrq	5
terminate_pd	ARELcf_TERrp	FTERcf	6
closed	FINIrq	AASSrq_INIrq	1
initialize_pd	AASScf_INIrp	FINIcf	4

V. CONCLUSION

Formal specification based analysis in its first step produces a formal description of the protocol from the Standard's semi-formal specification. Data structure definitions are obtained by applying to the ASN.1 specification in the standard our ASN.1-to-Estelle mapping rules. For a protocol like FTAM specifying functional units in individual modules gives a modular specification. The next step is the formal analysis of that specification. We used the analysis tool Contest–Estl to easily produce control and data flow graphs. Our graphs have six important properties that simplify test generation. In the third step we applied the data flow graphs to decompose the FTAM protocol into individual data flow functions. From those functions we again applied Contest–Estl to automatically derive unparameterized test sequences.

Our investigation brings up several questions that should also be studied. For example, test generation in TTCN form and addition of a parameter enumeration tool are the topics of further research.

REFERENCES

[1] A. Aho, A. Dahbura, D. Lee, and M. U. Uyar, "An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours," *IFIP PSTV VIII,* Atlantic City, NJ, June 1988, pp. 75–86.

[2] S. Aggarwal, K. Sabnani, and B. Gopinath, "A new file transfer protocol," *AT&T Tech. J.*, vol. 64, no. 10, pp. 2387–2411, Dec. 1985.

[3] G. V. Bochmann, M. Deslauriers, and S. Bessette, "Application layer testing and ASN.1 support tool," in *Proc. GLOBECOM '86.*

[4] E. Cerny, G. V. Bochmann, and A. Carrière, "Testing implementations of an application-level communication protocol," *Proc. Fault Tolerant Comput. Syst. FTCS-15,* June 1985.

[5] ISO/TC 97/SC 21, File transfer, access and management, DIS 8571, 1987.

[6] ISO/TC 97/SC 6, "Profile of abstract syntax notation one," IS 8824, 1987.

[7] ISO/TC 97/SC 21, "Estelle: A formal description technique based on an extended state transition model," IS 9074, 1988.

[8] ISO/TC 97/SC21/WG1, "OSI conformance testing methodology and framework parts 1, 2 &3," IS 9646, 1991.

[9] ISO/TC 97/SC21, "Application layer structure," DP 9545, Feb. 1988.

[10] K. Sabnani and A. Dahbura, "A new technique for generating protocol tests," in *Proc. IEEE 9th Data Commun. Symp.*, Sept. 1985, pp. 36–43.

[11] B. Sarikaya, G. V. Bochmann, and E. Cerny, "A test design methodology for protocol testing," *IEEE Trans. Software Eng.*, vol. SE-13, pp. 531–540, May 1987.

[12] B. Sarikaya, B. Forghani, S. Eswara, "An Estelle based test generation tool," *Comput. Commun.*, **Nov.** 1991.