Formal specifications for protocols: Issues and experiences

Gregor v. Bochmann (University of Montreal),
Luigi Logrippo (Ottawa University) and
Behcet Sarikaya (Concordia University, Montreal)

February 1990

Abstract

With wide-spread acceptance of the ISO-OSI reference model and its standardized protocols in the areas of computer communication and information exchange, formal specifications have become an area of active research and development. This paper surveys issues and recent developments obtained mainly from our undergoing research. The discussion includes four important aspects of the area: protocol design and specification languages, validation of the resulting design and specification, implementation development and finally conformance testing and implementation assessment.

1. Introduction

Communication protocols are the rules that govern the communication between the different components within a distributed computer system. In order to organize the complexity of these rules, they are usually partitioned into a hierarchical structure of protocol layers, as examplified by the seven layers of the standardized OSI Reference Model [Somm 89, Larm 88].

As they develop, protocols must be described for many purposes. Early descriptions provide a reference for cooperation among designers of different parts of a protocol system. The design must be checked for logical correctness. Then the protocol must be implemented, and if the protocol is in wide use, many different implementations may have to be checked for compliance with a standard. Although narrative descriptions and informal walk-throughs are invaluable elements of this process, painful experience has shown that by themselves they are inadequate.

The informal techniques traditionally used to design and implement communication protocols have been largely successful, but have also yielded a disturbing number of errors or unexpected and undesirable behavior in most protocols. The use of specification written in natural language gives the illusion of being easily understood, but leads to lengthy and informal specifications which often contain ambiguities and are difficult to check for completeness and correctness. The arguments for the use of formal specification methods in the general context of software engineering [Somm 89] apply also to protocols.

The following activities can be identified within the protocol development process. These activities can be partially automated if a formal protocol specification is used [Boch 87c].

(a) Protocol design: The protocol specification is developed based on the communication service to be provided by the protocol. The protocol also depends on the underlying (existing) communication service; e.g. the protocol may have to recover from transmission errors or lost messages if the underlying service is unreliable. The design process is largely based on intuition.

(b) Protocol design validation: The protocol specification must be checked (1) for logical consistency, (2) to provide the requested communication service, and (3) to provide it with acceptable efficiency.

(c) Implementation development: The protocol implementation must satisfy the rules of the protocol specification; the implementation environment and the user requirements provide additional constraints to be satisfied by the implementation. The implementation may be realized in hardware or software.

(d) Conformance testing and implementation assessment: The purpose of conformance testing is to check that a protocol implementation conforms to the protocol specification, that is, that it satisfies all rules defined by the specification. This activity is especially important for interworking between independently developed implementations, as in the case of OSI standards. The testing of an implementation involves three sub-activities: (1) the selection of appropriate test cases, (2) the execution of the test cases on the implementation under test, and (3) the analysis of the results obtained during test execution. The sub-activities (1) and (3) use the protocol specification as a reference.

In the context of the OSI standardization process [OSI 83, Larm 88], so-called formal description techniques (FDT) have been developed to be used for the writing of formal specifications of OSI protocols and services. They are called Estelle [Budk 87], LOTOS [Bolo 87] and SDL [Beli 89]. After arousing initially much expectations, their use is still quite limited in the work of the standardization committees and in the wider context of industrial protocol development (for a discussion, see for instance [Boch 89n]).

Much research and development efforts have gone into the development of tools to be used in relation with these languages. Some of these efforts will be mentioned in the sections below. Some of the results obtained from these efforts are quite interesting and useful. In the following sections, we try to give an overview of the important results relevant to the different phases of the protocol development cycle, and to highlight the issues that must be addressed to make the use of formal specifications for communication protocols a practical approach.

This paper is written from the perspective of our own ongoing research projects, which are partly funded by a collaborative grant from the Natural Sciences and Engineering Research Council of Canada. We do not pretend to give a general overview of this broad area, but hope to provide a critical discussion of our research and its relevance in the general context of protocol engineering.

2. Protocol design and specification languages

2.1 Formal description techniques and their use

The purpose of the two FDTs that most concerns us in this paper, Estelle and LOTOS, was to support OSI's standardization process. The third comparable FDT, SDL, was developed at first as a specification language for telephony, and was enhanced to include protocol specification concepts for OSI at the same time that Estelle and LOTOS were developed.

Since the purpose of these languages was OSI specification, they should have been available much before the beginning of OSI standardization. What happened, instead, was that their design was started at the same time as OSI standardization, and they became

available only when the standards for the first six layers of OSI were substantially completed. This simple chronological problem explains partly the relatively low level of acceptance of these FDTs in the OSI context.

OSI standards were developed without using the FDTs, by the usual well-established informal specification methods: English text, combined with various types of diagrams and semiformal notation [Boch 89d]. Subsequently, small groups of experts set about to specify some OSI layers, and as of today the following layers have been specified by using the FDTs: A draft specification of the transport protocol has been developed in an earlier version of Estelle, however this project has been abandoned within ISO. This work has been completed within a SEDOS project in Europe [Diaz 89]. Specifications in LOTOS have been completed in the form of ISO Technical Reports for the Session protocol and service [ISO/IEC/TR9572 and ISO/IEC/TR9571]. Work is continuing, although at a slow pace, on Technical Reports for LOTOS specifications of the Transport protocol and service [Working Draft SC6/WG4 N421 and ISO/IEC/DTR10023]. As far as we know, no firm decisions have been taken yet to undertake the formal description of other layers, or to use the FDTs in the standards, such as application layer standards, that are currently being developed.

Experts who have used the existing formal specifications have reported that they are difficult to read. No real attempt has yet been made to use such specifications for other purposes, such as test case generation or verification. It is important to remember, however, that the complexity of these specifications is largely due to the inherent complexity of the described protocol standards, which are the result of many compromises. The use of FDTs during the standardization process could be helpful for avoiding such unnecessary complexity, since the complexities would immediately become apparent in the formal specifications of the standard proposals.

In order to provide a readable and reasonable complex example of a protocol specification, we have developed specifications in Estelle and LOTOS of a simplified class 2 Transport protocol [Boch 90]; a sketch of a specification in SDL is also given. The similarity of these different specifications provides some basis for the comparison of the three specification languages.

We believe that the following factors have a strong impact on the acceptance of formal methods in the area of protocol development, and similarly in the more general context of software engineering:

(1) User training: Assuming interest to learn a new specification language, it is important to provide adequate learning material. Tutorials for Estelle and LOTOS have only been available recently. Most complete example specifications of OSI protocols have been considered difficult to understand by the layperson, while some of the pedagogical examples, e.g. [FDT GL], have been considered irrelevant.

(2) Intuitive language features: It is very useful if the basic language features are easily understandable by the layperson. Also the use of graphic representations facilitates the initial acceptance of the language because of its intuitive flavor, especially for smaller specifications (although graphics often becomes cumbersome for larger specifications).

Some examples of such features are FSM diagrams, entity-relationship diagrams for describing database structures, and inheritance relations in object-oriented languages. Graphic representations in SDL and the table-oriented structure of TTCN also provide an easy initial access to these languages. Currently, a graphic representation for LOTOS is being standardized.

(3) Relation between formal and informal specifications: Even the best formal specification will not replace an informal description of the specified system. The informal description will probably always remain the easier part to understand by the (human) designer and implementor. A straightforward relation between the informal and formal specifications will provide for easy cross-referencing between the two descriptions and promote the integration of the information provided by the two descriptions.

(4) Wide applicability: The FDTs seem to be applicable to other areas, in addition to the area of communication protocols; however, it is not clear how easily they can be adapted for writing object-oriented specifications. A wide applicability is an advantage, since the costs for development of support tools could be shared for a wide user community.

(5) Simple tools: Corresponding to intuitive language features (point (2) above), the functions provided by support tools should be intuitively easy to understand. In addition, it seems that the provision of a simple tool, possibly restricted in its functions, is better than the provision of a general tool which is difficult to use. For example, the reachability analysis tools based on FSM models have been found quite useful, although they are based on a restricted model and are not able to provide in practice a full analysis including interaction parameters.

(6) Integration into the general software/hardware development life cycle: Specifications are not used alone; as explained in Section 2.1, they are used throughout the protocol development cycle. Therefore the methods and tools related to formal specifications must be integrated with the other methods and tools used in the general software CASE or hardware CAD environment.

We expect that increased and more effective use of FDTs will result from increased knowledge of the languages in the community of protocol designers. It can then be expected that the specifier will start using FDTs early in the product or standard design phase, and the formal description will suit the product or standard much better than with the current "after-the-fact" methodology.

In the long term, as experienced with the FDTs increases, it will become more clear what are the desirable characteristics of such languages. Enhancements will be included in existing FDTs, and eventually better FDTs will be produced. However, judged by the example of what happened for programming languages, this evolution can be expected to be neither rapid, nor straightforward.

2.2 Example specifications

The results that we have had in our own work of application and enhancement of the existing FDTs have been encouraging and support the above conclusions. Following is a brief discussion of some of these experiences.

2.2.1 An industrial application of LOTOS

Gandalf is a Canadian supplier of communication equipments. Many types of standardized and proprietary protocols to coordinate and facilitate communication are an integral part of Gandalf's products. Informal or semiformal methods, such as various mixtures of English text, finite state machines, message diagrams, and program code fragments have previously been used to specify the protocols Gandalf has developed. These methods have resulted in errors, omissions and ambiguities being discovered during implementation and have

increased the cost of integrating new products and features into existing product lines. The errors appeared to be due at least in part to the imprecise semantics of the specification techniques mentioned above.

As a consequence, Gandalf wished to evaluate more formal specification methods. A pilot project was set up to this effect. In the first phase of this project, the language LOTOS was used for the specification of an existing Gandalf protocol, to see whether the language could be easily learned by protocol experts and to evaluate it as a specification method. In the second phase, LOTOS would then be used in the design of a new product. This application is, as far as we know, the first industrial application of LOTOS in North America.

The Gandalf project now has successfully passed the first phase. Gandalf specialists have learned LOTOS and have used it in order to specify certain aspects of a major company protocol, part of a distributed data PBX. It was not a simple protocol to specify, since it involved both software and hardware aspects, and since it had been developed incrementally over the years. LOTOS was found adequate for describing the architecture of the data switch both "in the large" (macro system design) and "in the small" (micro-system design). The project is now at the beginning stages of the second phase, where LOTOS are being applied in the development of a new protocol.

From this project, we are gaining valuable experience on the applications of LOTOS in an industrial environment [Logr 89]. In collaboration with Gandalf, we are planning to develop methodologies for the use of LOTOS through the whole life cycle of product development. Software tools and industrial methods will have to be developed, to help in the production and maintenance of specifications, in the development of implementations from specifications, and in the derivation of test suites for implementation. These are all current research topics, in which we hope that the continuation of the Gandalf project will provide contributions.

2.2.2 Different styles of LOTOS specifications

As most languages, LOTOS allows for a variety of different styles of using the language. The following styles have been identified in [Viss 88].

- Monolithic, where the specification is represented as a tree of choices. The parallel composition operator is not used in this style.
- State-oriented, where states are explicitly represented by using state variables.
- Constraint-oriented, where the specification is designed as the parallel composition of processes where each process separately enforces some set of behavior constraints.
- Resource-oriented, where parallel processes are chosen in such a way as to correspond to implementation modules.

The constraint-oriented style appears to be the most frequently used in the published OSI specifications mentioned above. It is our experience, however, that while this style works very well for specifying certain types of well-structured systems (such as the Transport Service or the telephone systems described in [Logr 90]), it leads to hard-to-

read specifications when applied to complex systems where many cases have to be considered.

A researcher in our group has produced a complete LOTOS specification for the X.25 link layer protocol, LAP-B, including all options [Guer 89, Guer89a]. The specification consists of about 2 300 LOTOS lines. This specification constituted a useful exercise in LOTOS styles. In this specification, a mixture of resource-oriented, state-oriented, and monolithic styles was used. While unfortunately the unavailability of differently structured LAP-B specifications makes it impossible to make comparisons, our specification appears to be fairly readable and has a reasonable degree of correspondence with the standard LAP-B specification. Furthermore, as mentioned in Section 5, the specification was successfully used for the derivation of test cases.

In view of these observations we believe that more experimentation is necessary on the question of LOTOS styles and their suitability in different contexts.

2.3 Semiformal description techniques in OSI

The semiformal techniques ASN.1 and TTCN are much more used in OSI than the FDTs. For the application of FDTs, it is therefore important to consider their relation with these semiformal techniques. As discussed below, there are still many unresolved issues.

2.3.1 ASN.1 (Abstract Syntax Notation One)

This is a notation for describing data structures [ASN.1, Neuf 90], similar to the data type definitions available in programming languages such as Pascal or ADA. It is applied to the description of OSI Application layer protocols, where it is used for the definition of the protocol data units (PDU's that is, the messages exchanged between different protocol entities). The notation includes a number of predefined data types, such as integers, reals, booleans, bit strings, octet strings and various kinds of character strings. It also allows the definition of composed data types, such as groups of elements (called SEQUENCE, corresponding to "record" in Pascal), a list of identical types (called SEQUENCE OF), a type of alternatives (called CHOICE, corresponding to Pascal's variant records), a TAG defining a code to distinguish between different alternatives, and others.

The main reason for the success of ASN.1 as specification language is probably the fact that it is combined with a standard encoding scheme for PDU's [ASN.1 C] which has been adopted for OSI Application layer protocols. Based on the information contained in the ASN.1 definition of the PDU structure, this scheme completely determines the PDU encoding, and can be used for implementing the encoding and decoding functions in a systematic manner, possibly automatically.

ASN.1 does not have the scope of an FDT, since it only describes data structures. However, it directly relates to the corresponding data structure definition facilities of the respective FDTs. Different scenarios for the interworking between ASN.1 and an FDT can be considered [Boch 89h]: (1) translation from ASN.1 into the corresponding FDT

language constructs, (2) addition of the ASN.1 notation to the FDT, or (3) replacement of the corresponding FDT language constructs by the ASN.1 notation. The translation approach has been explored for Estelle [Boch 90c, Barb 89] and LOTOS [Boch 89h]. A similar approach could also be used for SDL. The approach allows the combination of tools for PDU encoding and decoding, based on ASN.1, with tools for implementation/simulation of Estelle, SDL or LOTOS specifications.

Most ASN.1 concepts can be easily translated into corresponding FDT concepts. However, the ASN.1 list structure (i.e. SEQUENCE OF) leads in Estelle to partly implementation-oriented data structures.

The experience with LOTOS [Boch 89h] shows that the abstract data type notation ACT ONE [Ehri 85], used in LOTOS, is very cumbersome for the description of simple data structures because of the lack of suitable notations. Unless an abbreviated notation for data

structures (see proposals in [Scol 87, Boch 89h]) is introduced into LOTOS, the definitions of data structures in LOTOS are of the order of 8 times longer than the corresponding definitions in ASN.1. This leads to very lengthy and unreadable descriptions of the PDU data structures.

2.3.2 TTCN (The Tree and Tabular Combined Notation)

This notation is relatively recent, and has been developed for the description of test cases for OSI conformance test suites [OSI C3]. As its name indicates, the language includes several different notations. The overall organization of the language is in terms of a collection of tables defining different aspects of a test case, such as service primitives, PDU's and their parameters, order of interactions, and constraints on parameter values. The interaction ordering is defined in terms of a conceptual tree where each branch represents a possible execution order. In addition to the tabular notation, a linear form of TTCN is developed for the exchange of test cases in machine-readable form. The ASN.1 notation can also be used for certain aspects of test descriptions.

When the need for a notation for OSI test cases arose around 1985, the responsible standardization subcommittee was not ready to adopt one of the developing FDTs for this purpose, which in the authors' opinion would have been a reasonable choice. Instead, a new language TTCN was developed, which seems in many respects quite "ad hoc". Its semantics is defined largely informally. In order to formally relate the defined test cases to the corresponding protocol specification, a definition of its semantics in terms of one of the FDTs would be desirable [Sari 88f].

As in the case of ASN.1, the application of FDTs in the context of OSI, requires a translation between TTCN and the FDT used for the protocol specification. As discussed in Section 5, translation from the FDT into TTCN is required when test cases are developed from the formal specification of the protocol. The validation of test cases in respect to a formal protocol specification also requires a translation. In some of our work we have explored the use of an intermediate formal language (so-called charts, see Section 5) which can provide a bridge between the different description techniques.

2.4 Object-oriented languages and other techniques

Object-oriented programming seems to be a buzz-word at the present time. Independently of its popularity, it seems that the object-oriented concepts of "object" and "inheritance" are in fact quite useful in the context of specification languages. Systems are conceived by humans in terms of "objects" which have an intuitive meaning, and the inheritance relations among the specified object classes provide a clear structure for specialization and generalization.

In the OSI standardization work on management of distributed systems and "Open Distributed Processing" (ODP), object-oriented description models are being used. For this purpose, two extensions of the ASN.1 notation are of particular interest:

(a) The notation for "remote operations" (ROSE) [OSI RO] which is used to define the operations which are provided by an object and can be invoked by other (remote) objects.

(b) A notation for defining object classes [OSI MO] including the concepts of object attributes and inheritance of properties among classes.

The writing of object-oriented specifications is not necessarily well supported by the FDTs [Cusa 89a, Blac 89]. In a research project in collaboration with BNR, we have developed an object-oriented specification language which combines certain aspects of LOTOS (e.g. rendezvous interactions) with inheritance and the view that "everything is an object"; in particular, no syntactic distinction is made between "data structures", abstract data types, and processes [Boch 89e]. We have also integrated certain concepts from (object-oriented) databases, which is important for real-time control applications. The intended initial application of this language is in the network management area; the development of a larger trial specification is in progress. A description of the architecture of the OSI reference model is discussed in [Mond 90].

At this point, one may wonder whether it is a good idea to develop a new language, instead of using an existing one. In particular, high-level logical descriptions can often be interpreted by logic programming languages, such as Prolog. Prolog has been found useful for the development of protocol validation and test selection tools [Sidh 88a], although as a specification language for communication protocols, it has certain disadvantages [Boch 85]. It has the advantage of being a general purpose language for which many tools already exist. We are presently experimenting with the use of Prolog for a knowledge-based approach to the specification of OSI application layer protocols, which facilitates the creation of a simple user interface for browsing through different parts of the specification, and the use of the same specification for simulation, test suite development and test result analysis [Boch 89j]. We also try to identify a process which leads in a systematic manner from the informal specification documents to the formal specification in Prolog.

3. Design validation

Several different techniques are known to do design validation. Most of them presuppose that the protocol is at least partly specified in some formalism. For example, some techniques content themselves with validating the general structure of the system. In such cases, only the aspects that should be verified need to be formally specified.

Other techniques attempt to verify all aspects of the system. In this case, the system needs to be entirely and rigorously specified in some FDT. No technique in this latter category, however, comes even close to validating real-life systems. Complete proofs have been developed for "toy" examples such as the well-known alternating bit protocol. Larger proofs become quickly extremely cumbersome.

However, if the entire specification is executable or may be simulated in some appropriate environment, it is possible to validate it by executing tests, that is, by performing and validating certain selected execution paths [Zafi 80]. This method will not cover all possibilities, however, by selecting appropriate paths to be tested, many errors can be found without too much effort.

Various execution and simulation environments have been developed for FDTs and other specification languages. In the case of Estelle and SDL, where tools for the semi-automatic generation of implementation code exist (see for instance [Boch 87c]), these

tools may also be used for the validation of the specifications. Certain environments also provide for the simulation of performance parameters [Boch 87e] and/or provide facilities for observing the interactions exchanged between the different components of the specified system [Jard 85b]. It is also possible to combine the execution with the validation of predicates which are specified by the designer and should be satisfied in any execution of the system [Graf 89].

For the specification language LOTOS, two simulators have been developed independently [Eijk 88, Guil 88]. They both allow to execute the system in a step-by-step fashion, with the user providing the role of the environment and resolving nondeterministic choices. In an ongoing project [Saba 90], the simulation approach is used to validate a simplified specification of the Transport protocol [Boch 90] in respect to the OSI Transport service [OSI TS] written in LOTOS. A system of two interacting Transport entities and an underlying Network service are simulated by the LOTOS interpreter, and the resulting sequence of service primitives is validated against the service specification by using a trace analysis tool TETRA [Bell 89] initially developed for the automatic analysis of conformance test results (see Section 5.5).

In contrast to the simulation approach, the well-known reachability analysis explores systematically all possible execution paths. However, for realistic protocols, this technique is only applicable to an approximated specification restricted to a finite state machine model. In the case of LOTOS, certain tools only consider the so-called "basic LOTOS" subset which ignores interaction parameters. Related to CSP [Hoar 85] and CCS [Miln 80], this subset allows the application of similar exhaustive validation techniques, if the system is not too large [Najm 89] (also [Shir 89]).

A method in between reachability analysis and testing through simulation has been explored with a tool which provides for the generation of partial execution trees for LOTOS specifications including symbolically evaluated interaction parameters. This facility performs a sort of "eager evaluation" where all possible execution paths are explored, with symbolic values in lieu of the values to be provided by the environment. The symbolic tree evaluator has been enhanced to identify certain situations of repeated behavior [Guil 89]. A further step is a tool which provides the monolithic LOTOS specification corresponding to the tree [Quem 89, Ashk 90].

In the case of simple specifications, where the entire execution tree or monolithic specification can be calculated, these tools will provide a complete analysis of all possible behaviors of the specification. For example, deadlocks will be explicitly revealed by the tool, while other undesirable behaviors would be revealed by inspection. With larger specifications, however, the usefulness of this tool is greatly diminished by two factors. First of all, most such specifications branchout too quickly for the whole execution tree to be completed. Even worse, tests involving symbolic values cannot usually be evaluated to "true" or "false", thus a great number of unfeasible paths is generated. This problem can be reduced or limited by the use of appropriate specification styles [Guer 89a, Guer 89].

For the comparison of LOTOS specifications, several notions of equivalence may be used. Much work has been based on bi-simulation equivalence (e.g.[Miln 80, Miln 89,

Brin 88, Najm 89, Shir 89]). However, the notion of testing equivalence, which is related to the "traces and refusals" of CSP [Hoar 85], seems to be more realistic, since it is based on whether two processes can be distinguished by executing testing experiments. Using such methods, it is possible to prove directly properties of traces (i.e. sequence of actions), such as: "the number of interactions at gate g will always exceed by one the number of interactions at gate h". Properties of this type are frequently stated for processes. Corresponding proof rules for LOTOS specifications have been developed [Gall 89]. Important immediate applications of verification principles have been found in the area of equivalence-preserving transformations. In LOTOS, a theory is being developed, supported by software tools, on how a specification written in one style can be transformed into an equivalent one written in another style. For example, one may wish to transform a specification written in an abstract, implementation-independent style, into another one that can be easily coded [Ashk 90, Quem 89, Viss 89, Eijk 89a, Leon 89].

Unfortunately by using these methods, it has only been possible so far to prove the correctness of small didactical examples. Semi-automated verification tools are being devised, but on the basis of what has been seen in the case of verification methods for functional programming languages, it can be assumed that the power of such tools will not go beyond proofs of specifications of a few hundred lines. This is an area where research is very active, however, some fundamental problems exist, by which progress can be expected to be slow.

4. Implementation development

The requirements to be satisfied by a protocol implementation include the protocol specification and usually additional constraints which are particular to the implementation project. These additional constraints may define such questions as "how does the implementation react to unexpected (invalid) user interactions?", "how many simultaneous connections should be supported?", or "what should be the performance of the implementation?". Based on these requirements, the implementation is usually developed in several steps of refinement using the standard software or hardware design and implementation methods.

In the case that a formal protocol specification is used, the additional constraints may be defined in the same formal specification language, and, depending on this language, may be translated into implementation code. The semi-automatic implementation based on Estelle specifications has been used in a number of cases. This topic has been covered relatively well in the existing literature [Boch 87h, Sidh 89]. Similar translation approaches can be used with other languages. In the case of LOTOS, it seems reasonable to impose certain language restrictions for the translation [Mana 89].

In the case that a multi-layer protocol structure is distributed over several operating system tasks or computers, that a communication service among several computers is to be simulated, or that a specified application is to be implemented in a distributed environment, it would be useful to have a uniform distributed implementation environment. In such an environment, a single system specification including multiple

processes/modules could be automatically implemented by allocating each process/module to a particular operating system task or computer.

In the case of Estelle and SDL specifications, such a distributed implementation is relatively easy to obtain, since the processes/modules communicate by message passing, which is easily mapped on a message transmission communication facility available between the tasks/computers [Jard 89]. (For Estelle, the implementation is simplified if all dependents of a system activity or system process reside in the same task/computer).

In the case of LOTOS the situation is more difficult because of the rendezvous nature of interactions between the processes [Quem 89, Karj 88]. Several distributed algorithms for the implementation of rendezvous interactions between remote systems are known. The situation for LOTOS is particularly complex because more than two processes may participate in an interaction, and the processes involved cannot be determined statically. We have shown [Boch 89c] that a tree-oriented execution model for LOTOS can be combined with a virtual-ring rendezvous protocol, in order to obtain a scheme for the distributed implementation of LOTOS specifications. Present work is aimed at building a prototype implementation of a distributed LOTOS simulation environment, using multiple instances of an existing LOTOS interpreter [Logr 88], and at the realization of complete and fair interpretations in such a distributed environment.

As mentioned in Section 2, the integration with ASN.1 is an important issue for any specification language to be used for the OSI application-layer protocols. Some of our recent research projects were related to this question. While certain authors suggest the enhancement of existing specification languages with the data type notations of ASN.1, we have explored the possibility of translating the ASN.1 definitions of the PDU structures into the corresponding data type definitions of Estelle [Boch 90, Barb 89] and LOTOS [Boch 89h].

The translation for Estelle is relatively straightforward, except for the SEQUENCE OF construct which is translated into a list structure implemented by pointers. In order to obtain tools for supporting protocol implementations for this approach, we have built an ASN.1 translator and have adapted an existing ASN.1 tools which generates ASN.1 encoding and decoding routines [Yang 88] with an Estelle compiler [Este 87b] in such a manner that the C-code generated by the different tools is compatible and can be combined into a single protocol implementation.

The encoding and decoding of ASN.1 PDU's is a relatively tedious task that can be implemented in specialized hardware in order to obtain increased efficiency [Bilg 90]. It is to be noted, however, that the encoding/decoding modules (in software or hardware) are closely related to the data structures that are used within the implementation of the protocol implementation.

Sometimes the protocol specification is written in a form which is either not directly implementable by the translation approach discussed above, or would lead to implementation code which is not sufficiently efficient. In such cases, it is possible to first derive a more implementation-oriented description from the given specification, and then to apply the automatic translation approach to the implementation-oriented

description. Much research has been done in the related general area of program transformations [Baue 79], where one wants to assure that the transformed program is equivalent to the originally given program or specification. Research along these lines is going on in relation with the LOTOS language (see Section 4), however, this is a difficult problem and it is not clear how useful this approach will be in the future.

5. Conformance Testing and Implementation Assessment

In the testing area, formal specifications of the protocol can be used for deriving test suites, validating manually developed test cases, and for analyzing test results. The use of TTCN for the description of OSI conformance test cases leads to the need for translation between various languages. The following subsections comment on related issues.

5.1 Test suite development

There are two main approaches for the design of a conformance test suite: manual design and semi-automatic design based on formal description of the protocol. There are several test suites that are manually developed and are being standardized [ISO 8882, and others]. We will concentrate on semi-automatic design from validated formal specifications and describe the techniques and tools available presently. Most of the literature deals with test generation from finite state machines [Sari 82, Sabn 85, Vuon 89, Fuji 90]. Test sequence derivation from FDT specifications has also received attention. First a comprehensive methodology based on Estelle has been developed [Sari 87]. Recently LOTOS based test derivation research has been active [Brin 88, Weze 89, Guil 88, Sari 89e, Trip 89a].

Estelle based test design methodology first applies syntactic transformations on the protocol specification and puts the specification into a form called normal form in which transitions contain single paths and local procedures/ functions are in-line replaced. The normal form transformations are applied to all the modules in case of modular specifications. The next step is to apply control flow and data flow abstraction on the specification. In the control flow abstraction the specification is treated as a collections of interconnected FSMs. This representation is easily obtained from the normal form transitions. The data flow abstraction represents the action part of the transitions as a data flow graph in which abstract service primitive (ASP) and protocol data unit (PDU) parameters become input/ output nodes and the data flow on these are determined from the assignment statements.

The FSM model of the specification makes it possible to use certain techniques to derive test sequences such as transition tours. In the case of modular specifications, transition tour generation considering the internal queues is rather complicated [Forg 90]. The resulting test sequence gives a control flow coverage of all the transitions. State recognition based techniques such as W- and D- methods could also be used; but their application to protocols seems to be controversial due to the fact that the choice of the state recognition sets/sequences requires the consideration of the context variables, rather than just the major state. There are also problems associated to incompletely defined FSMs [Vuon 89]. Since the transition tour coverage does not consider enabling conditions, some of the resulting paths are infeasible. It seems that avoiding these infeasible paths will remain in the domain of human intelligence.

The dataflow graph obtained can be used to derive the dataflow associated with "protocol functions". Since protocol functions are related to the understanding of the protocol, the derivation of them can only be done by the user.

A tool has been developed implementing this methodology. The control and dataflow graphs are visually displayed on a workstation. The test generation module assists the user in eliminating the infeasible paths. The dataflow graphs can be partitioned and named by the user. Then the test generation module derives the test sequences needed to completely cover the dataflow in each function. This way the resulting test sequences completely cover the control and dataflow in the specification. The dataflow coverage criteria is simple. A more involved dataflow coverage considering the definition and use of the context variables is studied [Ural 87a]. Presently it is not yet determined which coverage technique will yield better test sequences. The tool implementing the methodology has been applied to generate test sequences for transport protocol [Forg 89], FTAM [Barb 89], ISDN LAPD [Sari 89d] and ISDN Network layer [Amal 90].

In the case of LOTOS specification, the derivation of test cases is complicated due to the parallel composition and the nondeterministic nature of the specifications. Three different approaches to deriving test cases have been described in the literature.

(1) Canonical test processes: Given a basic LOTOS specification S of a protocol, an "inverse" specification T(S) can algorithmically be developed such that when an IUT is parallelly composed with T(S), it will apply all the test cases to establish the conformance of IUT to S. The method is named CO-OP after its main components, the

sets Compulsory and Optional behaviors [Weze 89]. The T(S)obtained is a choice of all the test cases, except robustness tests. It is shown that T(S) is equivalent to the canonical tester defined earlier in [Brin 88]. It is also shown in [Weze 89] that if S is a parallel composition S: = B1 |||| B2 (contrary to intuition) we cannot simply obtain T(B1) and T(B2) and then parallelly compose them to get T(S).

A canonical tester is not intended for practical testing because T(S) can only be derived from basic LOTOS, although some efforts to extend the approach have been reported [Tret 89]. T(S) is another specification in LOTOS usually with an infinite behavior, and therefore it is difficult to derive a finite test suite.

(2) Interpretation of specification: Full LOTOS can be symbolically executed using an interpreter. Symbolic execution derives execution paths that can be used as test sequences [Guil 88]. Although such trees or expansions are usually partial, they can still contain enough information to be used in test case generation. In [Guil 89] this idea was applied in order to generate test cases for LAP-B based on the specification mentioned in Section 2.2.2. From the execution trees obtained from the interpreter, a FSM model was manually derived. Then traditional test selection methods for FSM's could be applied. The authors have not yet considered the dataflow aspects with the interpretation approach.

(3) Analysis of control and dataflow: Applying an extended FSM interpretation to LOTOS semantics, a LOTOS specification can be converted into an equivalent chart [Miln 84]. A translation algorithm of a subset of LOTOS into charts is reported [Karj 88]. This algorithm could be extended to full LOTOS but in some cases, i.e. parallel activation of an unknown number of connections, the chart obtained is infinite. Practice indicates that the chart construction algorithm leads to a state explosion even for medium size specifications such as [Boch 90] due mainly to state machine interpretation of parallel behavior. However using some heuristics, the number of states could be reduced to reasonable size. The chart obtained represents the overall behavior of the entity specified, i.e. the control and dataflow in the specification. The dataflow can be abstracted out and represented in graphical form. These graphs are similar to the graphs obtained from the Estelle specification of the same protocol. The complete traversal of the chart yields the test sequences (may-tests), considering nondeterminism (the "i" events) the tests can be converted into test cases (must-tests) [Trip 89a]. Another approach, also considering the control and dataflow of LOTOS specifications is reported in [Sari 89e].

5.2 Test specification in TTCN

It is important to note that the OSI conformance test cases are described in a newly developed notation called TTCN [OSI C3]. Since specifications written in this language are presented in the form of tables, screen-oriented editors have been developed, although a linear form of the language has also been defined.

In the case of semi-automatic test suite design, a translation from the language of the protocol specification into TTCN is required. If one wants to validate a TTCN test case

in respect to a protocol specification, the inverse translation is necessary. Together with ASN.1, which can be used as part of TTCN, there are many translation issues. In addition, there are many proprietary test description languages for which test execution environments exist. Test cases written in TTCN will therefore often be translated into such languages for execution. The use of an intermediate language, such as the charts mentioned above, has been proposed as a common ground for all these translations. However, it is not clear what the best approach to this proliferation of specification languages is.

5.3. Test case validation

The tests obtained with the semi-automatic test design tools are expected to be correct (if the tool has been sufficiently debugged). However, manually developed test cases can be

expected to contain certain errors, due to the large amount of information that must be considered in their design. They can be validated against a formal specification of the protocol, using automated tools.

First, a test case can be validated by checking certain static conditions by a TTCN compiler. Second, the semantics of the test case must be validated against the protocol specification. In particular, each path in the test case leading to a PASS (FAIL) verdict must be shown (not) to be accepted by the formal specification. Additional validation may concern time-out behaviors or buffer sizes.

Two approaches to the automatic validation of test cases have been explored in our projects. After translation into LOTOS behavior expressions, the test case can be automatically compared with a given protocol specification written in LOTOS [Boch 89j]. The tool is an extension of what is described for test result analysis in Section 5.5. Another approach is the translation of both the test case and the specification into charts (see above) and their comparison at this intermediate level [Naik 90].

5.4 Test case adaptation and execution

Most standardized test cases, as well as most tests derived from formal specifications, usually can be classified as generic test cases for the local single-layer test architecture. In order to executed them with a test system using a different test architecture, such as the distributed test architecture [Rayn 87], they must be adapted. However, these necessary modifications are relatively straightforward in most cases.

The test cases must also be adapted to the implementation parameters of the protocol implementation, which are stated in the so-called PICS (protocol implementation conformance statement) and the PIXIT (protocol implementation extra information for testing). Certain values of test parameters are not specified in the generic test cases. They may be chosen during the test execution within the bounds provided by the implementation parameters.

Different approaches can be taken to execute test cases written in TTCN. Most systems for TTCN implementation support translate the test cases into implementation languages, such as C or specialized test description languages for which an execution environment already exists. The translation into Estelle can also be considered [Eswa 90], since good implementation environments for Estelle exist, and this language seems suitable for the specification and implementation of complete test systems [Linn 88]. It is to be noted that the use of ASN.1 in the test cases for Application layer protocols implies that ASN.1 support must be included in the test execution environment.

5.5. Test Result Analysis

Execution of each test case on an IUT will result in a conformance log containing the events that occurred with time stamps. Trace checking and test result analysis refer to the activity in which the conformance log (trace or test results) is analyzed manually, or by using the formal specification of the protocol [Boch 89m, Boch 89j] and a verdict of pass

or fail is produced. Since TTCN test cases explicitly indicate the pass and fail cases in the definition, a test system executing TTCN test cases contains the result analysis already in the form of the test cases.

An automatic test result analysis in respect to the protocol specification is useful in the following situations:

(a) In the case that the IUT is subjected to ad hoc or random tests, for instance during debugging, or for complementing the standard conformance tests.

(b) For arbitration testing. This involves two or more systems that have already been tested individually, and which nevertheless turn out to have problems interworking. The testing architecture includes the tester passively observing the PDU's exchanged between the different systems. The tester includes a trace analysis module which checks the observed trace in respect to the specifications of all the systems and will notify any detected error.

(c) For validating the defined test cases. A suite of test cases for a given protocol can be very voluminous. Since most test cases are developed by informal methods, they may contain errors, that is, wrong verdicts. Automatic trace analysis can be used to check the verdicts of test cases with the formal specification of the protocol.

6. Concluding remarks

In our research we have used the three general purpose formal languages called LOTOS, Estelle and SDL, the data structure description notation called ANS.1 and the test specification language called TTCN for protocol design, validation, implementation development and assessment. While our research concentrated on LOTOS and Estelle it seems that presently ASN.1 and TTCN are more heavily used in industry. We discussed several ways in which heavier use of FDTs could be accomplished: developing easy-to-use tools, support for object-oriented techniques, pilot projects and so on.

There is a need to research towards arriving at a stage where protocol development activities could be supported with an integrated set of tools. The need to design validated specifications arouses the need to incorporate the PICS and PIXIT information in the formal specification, this in turn might require modifications in the languages. For example there is a need for an abbreviated notation for LOTOS data structures. It is interesting to investigate incorporation of PICS and PIXIT processing in test generation. Regarding ASN.1, it could possibly conclude that the best way to handle ASN.1 is by treating it as a tree structured data representation sublanguage instead of translating it into other languages. Finally, we point to the need for implementing protocols in parallel environments and by hardware.

Acknowledgements:

We would like to thank all our students and research associates who contributed to the experiences discussed in this paper, and are mentioned in the references. We would also

like to thank Pierre Mondain-Monval, Daniel Ouimet and Lucie Lévesque for helping us to put together the final version of this paper. This work was supported by a strategic research grant of the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[ASN1 C] ISO IS 8825, "Information Processing - Open systems Interconnection - Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)".
[ASN1] ISO IS 8824, "Information Processing - Open systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)".
[Amal 90] M.Amalou, G.v.Bochmann, publication departementale, 1990.
[Ashk 90] P.Ashkar, "A symbolic evaluator for LOTOS applications", M.Sc. Thesis, University of Ottawa, 1990 (in preparation).
[Barb 89] M.Barbeau, G.v.Bochmann, "Experience with automated verification tools: Application to discrete event systems", Prel. Proc. of Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, June 1989 (due to an error, the paper is not included in the final proceedings published in Springer LNCS).
[Baue 79] F.L.Bauer, et al., "Towards a wide-spectrum language to support program specification and program development", Lecture Notes in C.Sc. 69, Springer Verlag, 1979, pp. 543-552.
[Beli 89] Ferenc Belina, Dieter Hogrefe, "The CCITT-Specification and Description Language SDL", Computer Networks and ISDN Systems, Vol. 16, pp.311-341, 1989.
[Bell 89] O.Bellal, "Analyse automatique de résultats de tests appliquée aux protocoles de communication", Master's thesis, Dept. d'IRO, Univ. de Montreal, 1989.
[Bilg 90] M.Bilgic, B.Sarikaya, "An ASN.1 encoder decoder and its performance", Concordia University, 1990.
[Blac 89] S.Black, "Objects and LOTOS", FORTE'89, Vancouver, 1989.
[Boch 85] G.v.Bochmann, R.Dssouli, W.Lopes de Souza, B.Sarikaya, H.Ural, "Use of Prolog for building protocol design tools", Proc. 5-th IFIP Workshop on Protocol Specification, Verification and Testing, Toulouse, June 1985, North-Holland Publ., pp. 131-147.
[Boch 87c] G.v.Bochmann, "Usage of protocol development tools: the results of a survey" (invited paper), 7-th IFIP Symposium on Protocol Specification, Testing and Verification, Zurich, May 1987, pp.139-161.
[Boch 87e] G.v.Bochmann, D.Ouimet and J.Vaucher, "Simulation for validating performance and correctness of communication protocols", Techn. Report, Department d'IRO, Université de Montréal, 1987.
[Boch 87h] G.v.Bochmann, G.Gerber, and J.M.Serre, "Semiautomatic implementation of communication protocols", IEEE Tr. on SE, Vol. SE-13, No. 9, September 1987, pp. 989-1000 (reprinted in "Automatic Implementation and Conformance Testing of OSI Protocols", IEEE, edited by D.P.Sidhu, 1989).
[Boch 89c] G.v.Bochmann, Q.Gao, C.Wu, "On the distributed implementation of LOTOS", FORTE'89 (IFIP), Vancouver, 1989.
[Boch 89d] G.v.Bochmann, "Protocol specification for OSI", to be published in Computer Networks and ISDN Systems.

[Boch 89e] G.v.Bochmann, et al., "Object-oriented databases: Modelling and specification of applications in the field of network management", Report for research contract CRIM/BNR, April 1989.
[Boch 89h] G.v.Bochmann and M.Deslauriers, "Combining ASN1 support with the LOTOS language", Proc. IFIP Symp. on Protocol Specification, Testing and Verification XI, June 1989, North Holland Publ., pp.
[Boch 89j] G.v.Bochmann and O.Bellal, "Test result analysis in respect to formal specifications", Proc. 2-nd Int. Workshop on Protocol Test Systems, Berlin, Oct. 1989.
[Boch 89m] G.v.Bochmann, R.Dssouli and J.R.Zhao, "Trace analysis for conformance and arbitration testing", IEEE Tr. on Softw. Eng., Nov. 1989, pp.1347-1356.
[Boch 89n] G.v. Bochmann, "Formal methods for describing distributed systems: A discussion of the experience in OSI standardization", Proc. IFIP Working Conf. on Decentralized Systems, Lyon, Dec. 1989 (invited paper).
[Boch 90] G.v.Bochmann, "Specifications of a simplified Transport protocol using different formal description techniques", to be published in Computer Networks and ISDN Systems.
[Boch 90c] G.v.Bochmann, D.Ouimet, G.Neufeld, "Implementation support tools for OSI application layer protocols", to be submitted to Software Practice and Experience.
[Bolo 87] T.Bolognesi and E.Brinksma, "Introduction to the ISO Specification Language LOTOS", Computer Networks and ISDN Systems, vol. 14, no. 1, pp.25-59, 1987.
[Brin 88] E.Brinksma, "A theory for the derivation of tests", Proc. IFIP Symposium on Protocol Specification, Testing and Verification, Atl. City, 1988.
[Budk 87] S.Budkowski and P.Dembinski, "An introduction to Estelle: a specification language for distributed systems", Computer Networks and ISDN Systems, vol. 14, no. 1, pp.3-23, 1987.
[Cusa 89a] E.Cusak, S.Rudkin, C.Smith, "An object-oriented interpretation of LOTOS", FORTE'89, Vancouver, 1989.
[Diaz 89] M.Diaz, et al., "The formal description technique Estelle", North Holland Publ. 1989.
[Ehri 85] H.Ehrig and B.Mahr, Fundamentals of Algebraic Specifications 1, Springer Verlag, 1985.
[Eijk 88] P.H.J. van Eijk, "Software tools for the specification language LOTOS", Doctoral Thesis, Universiteit Twente (1988).
[Eijk 89a] P. van Eijk. "Tools for LOTOS Specification Style Transformation" to appear in S. Vuong, Formal Description Techniques II, North-Holland
[Este 87b] NBS, "User guide for the NBS prototype compiler for Estelle", Final Report, Report no. ICST/SNA - 87/3, Octobre 1987.
[Eswa 90] S.Eswara, T.Berriman, P.vanHoutte, B.Sarikaya, "Towards execution of TTCN", Technical report, Concordia University, 1990.
[FDT GL] ISO Tech. Report, "Guidelines for the Application of Estelle, LOTOS and SDL", DTR 10167, 1989.
[Forg 89] B.Forghani, S.Eswara, V.Koukoulidis, B.Sarikaya, "An Estelle based test generation tool for modular Specifications", FORTE'89, Vancouver, 1989.
[Forg 90] B.Forghani, "Automatic test generation in TTCN from Estelle", M.Sc. Thesis, Concordia University, 1990.
[Fuji 90] S.Fujiwara, G.v.Bochmann, F.Khendek, M.Amalou, A.Ghedamsi, "Test selection based on finite state models: A unified view", submitted to IFIP Symposium on Protocol Specification, Testing and Verification, 1990.

[Gall 89] M.S.Gallouzi, "Trace analysis of LOTOS behaviors", M.Sc. Thesis, University of Ottawa, 1989.
[Graf 89] S.Graf, J.-L.Richier, C.Rodriguez, J.Voiron, "What are the limits of model checking methods for the verification of real life protocols?", in Lecture Notes in Computer Science "Automatic Verification Methods for Finite State Systems International Workshop", G.Goos and J.Hartmanis (Eds.), Grenoble, France, Springer-Verlag, June 1989.
[Guer 89] D.Gueraichi, L.Logrippo, "Derivation of test cases for LAP-B from a LOTOS specification", to appear in the Proc. of the 2nd FORmal TEchniques Symposium (Vancouver, December 1989).
[Guer 89a] D.Gueraichi, "Derivation of test cases for LOTOS LAP-B specification", M.Sc. Thesis, University of Ottawa, 1989.
[Guil 88] R.Guillemot, M.Hay-Hussein, L.Logrippo, "Executing large LOTOS specifications", Proc. IFIP Symposium on Protocol Specification, Testing and Verification, Atl. City, 1988.
[Guil 89] R.Guillemot, L.Logrippo, "Derivation of useful execution trees from LOTOS by using an interpreter", in "Formal Description Techniques", (Ed.) K.J.Turner (Proceedings of the First International Conference on Formal Description Techniques Stirling, Scotland, 6-9 September, 1988), pp.311-325.
[Hoar 85] C.A.R.Hoare, "Communicating Sequential Processes", Prentice Hall, 1985.
[ISO 8882] ISO, "X.25-DTE Data link layer conformance test suite", TC97/SC6, ,DIS8802 Part 2, 1989.
[Jard 85b] C.Jard, R.Groz, J.F.Monin, "VEDA: a software simulator for the validation of protocol specifications", Proc. COMNET '85 (IFIP), Computer Network Usage: Recent Experiences, North Holland, Oct. 1985.
[Jard 89] .Jard, J.M.Jazequel, "A multi-processor Estelle to C compiler to experiment distributed algorithms on parallel machines", Proc. IFIP Symposium on Protocol Specification, Testing and Verification IX, Twente, 1989.
[Karj 88] G.Karjoth, "Implementing process algebra specifications by state machines", Proc. IFIP Symposium on Protocol Specification, Testing and Verification, Atl. City, 1988.
[Larm 88] K.G.Knightson, T.Knowles and J.Larmouth, "Standards for Open Systems Interconnection", McGraw-Hill, 1988.
[Leon 89] G.Leon, C.Delgado Kloss , G.Gonzalez, M.A.Ruiz, S.Marchena, L.Santos, J.Navarro, "ASDE: Design of a transformational environment for LOTOS", to appear in S.Vuong, Formal Description Techniques II, North-Holland.
[Linn 88] R.J.Linn and J.P.Favreau, "Application of Formal Description Techniques to the Specification of Distributed Test Systems", Proc. of INFOCOM'88, pp.96-109, IEEE, March 1988.
[Logr 88] L.Logrippo, et al., "An interpreter for LOTOS: A specification language for distributed systems", Software Practice and Experience, Vol. 18(4), pp.365-385, April 1988.
[Logr 89] L.Logrippo, T.Melanchuk, R.J.DuWors, "The algebraic specification language LOTOS: an industrial experience", University of Ottawa, Computer Science Department, report TR-89-35.
[Logr 90] M.Faci, L.Logrippo, B.Stepien, "Formal specification of telephone systems in LOTOS: the constraint-oriented style approach", will be published in Computer

Networks and ISDN Systems, preliminary version will appear in "Protocol Specification, Testing and Verification IX", North-Holland, 1989.
[Mana 89] J.A.Manas, T.DeMiguel and H.vanThienen. "The Implementation of a Specificatio Language for OSI Systems" In: P.vanEijk et al (Eds) The Formal Description Technique LOTOS. North-Holland 1989.
[Miln 80] R.Milner, "A calculus of communicating systems", Lecture Notes in Computer Science, No. 92, Springer Verlag, 1980.
[Miln 84] R.Milner, "A complete inference system for a class of regular behaviors", JCSS, 1984, pp.439-466.
[Miln 89] R.Milner, "Communication and Concurrency", Prentice-Hall, 1989.
[Mond 90] P.Mondain-Monval, "Object-oriented Model for the OSI Reference Model", submitted to IFIP Symposium on Protocol Specification, Testing and Verification, 1990.
[Naik 90] S.Naik, B.Sarikaya, "Static Validation of TTCN Test Cases" Technical report, Concordia University, 1990.
[Najm 89] E.Najm, "A verification oriented specification in LOTOS of the Transport Protocol", in "The Formal Description Technique LOTOS", P.H.J.van Eijk, C.A.Vissers, and M.Diaz (Eds.), pp.269-284.
[Neuf 90] G.W.Neufeld, "A Tutorial on ASN.1", to be published in Computer Networks and ISDN Systems, 1990.
[OSI 83] "Special issue on Open Systems Interworking", Proc. of the IEEE, Dec. 1983.
[OSI C3] ISO "DP 9646-3, Information Processing Systems - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN)".
[OSI MO] ISO/IEC JTC1/SC6 N5402 "Guidelines for the definition of managed objects" (Draft 1989).
[OSI RO] ISO IS 9072, "Remote Operations".
[OSI TS] ISO TC97/SC16, DP8072, "OSI - Transport service specification", 1983.
[Quem 89] J.Quemada, S.Pavon and A. Fernandez, "Transforming LOTOS specifications with LOLA. The Parameterised Expansion", in "Formal Description Techniques, K.J.Turner (Eds.), North-Holland, 1989, pp.45-54.
[Rayn 87] D.Rayner, "OSI Conformance testing", Computer Network and ISDN Systems, 14 (1987), pp. 79-98.
[Saba 90] Fayez Saba, "Validation en ligne de traces d'execution appliquee au protocole de Transport", M.Sc. thesis, Université de Montréal, summer 1990.
[Sabn 85] K.Sabnani, A.Dahbura, "A new technique for generating protocol tests", Proc. 9th Data Comm. Symp., 1985, pp.36-43.
[Sari 82] B. Sarikaya and G.v. Bochmann, "Some experience with test sequence generation for protocols", Proc. 2-nd Int. Workshop on Protocol Specification, Testing and Verification, North Holland, 1982, pp. 555-567.
[Sari 87] B.Sarikaya, G.v.Bochmann, E.Cerny, "A test design methodology for protocol testing" IEEE Trans. on SE, (May 1987), pp. 518-531.
[Sari 88f] B.Sarikaya, and Q.Gao, "Translation of test specifications in TTCN to LOTOS", Proc. IFIP Symposium on Protocol Specification, Testing and Verification, Atl. City, 1988.
[Sari 89d] B.Forghani and B.Sarikaya, "Automatic dynamic behaviour generation in TTCN format from Estelle specifications", Proceedings of the international workshop on Protocol Test Systems, Berlin (West), Germany, October 1989, pp.73-90.

[Sari 89e] B.Sarikaya and S.Lu, "A LOTOS based test design methodology", Technical report, Concordia University, 1989.
[Scol 87] ISO 97/21 N1540, "Potential enhancements to LOTOS", 1986.
[Shir 89] N.Shiratori, H.Kaminaga, K.Takahashi and S.Noguchi, "A verification method for LOTOS specifications and its applications", to appear in IFIP Protocol Specification, Testing and Verification IX, C.A.Vissers (Eds.), North-Holland, 1989.
[Sidh 88a] D.P. Sidhu and C.S. Crall, "Executable Logic Specifications for Protocol Service Interfaces", IEEE Transactions on Soft. Eng., Vol. 14, No.1, January 1988.
[Sidh 89] D.P. Sidhu and T.P. Blumer, "Automatic Implementation of Protocols", to be published in Computer Networks and ISDN Systems.
[Somm 89] Ian Sommerville, "Software Engineering", 3-rd ed., Addison-Wesley Publ. 1989.
[Tret 89] J.Tretmans, "Test case derivation from LOTOS specifications", FORTE'89, Vancouver, 1989.
[Trip 89a] P.Tripathy and B.Sarikaya, "Test generation from protocol specification", FORTE'89, Vancouver, 1989.
[Ural 87a] H.Ural, "Test sequence selection based on static dataflow analysis", Computer Communications, Vol.10, No.5, October 1987.
[Viss 88] C.Vissers, G.Scollo, M.v.Sinderen, "Architecture and Specification Style in Formal Descriptions of Distributed Systems", Proc. IFIP Symposium on Prot. Spec., Verif. and Testing, Atlantic City, 1988.
[Viss 89] C.A.Vissers, G.Scollo, M.v.Sinderen, E.Brinksma, "On the use of specification styles in the design of distributed systems", University of Twente, 1989.
[Vuon 89] S.T.Vuong, W.L.Chan, M.R. Ito, "The UIOv-method for protocol test sequence generation", proceedings 2-nd Int. Workshop on Protocol Test Systems, Berlin, Oct. 1989.
[Weze 89] C.D. Wezeman, "The CO-OP method for compositional derivation of conformance testers", Proc. IFIP Symposium on Protocol Specification, Testing and Verification IX, Twente, 1989.
[Yang 88] Y.Yang, "ASN.1-C Compiler for Automatic Protocol Implementation", Master Degree Thesis, Department of Computer Science, University of British Columbia, October 1988, 111pp.
[Zafi 80] P.Zafiropulo, C.H.West, H.Rudin, D.D.Cowan, and D.Brand, "Towards analyzing and synthesizing protocols", IEEE Tr. Comm. COM-28, 4 (April 1980), pp. 651-660.