FORMAL METHODS FOR DESCRIBING DISTRIBUTED SYSTEMS:

A DISCUSSION OF THE EXPERIENCE IN OSI STANDARDIZATION*

Gregor v. Bochmann

Département d'informatique et de recherche opérationnelle
Universite de Montreal, Canada

Abstract

Distributed systems are difficult to design and implement because of concurrent activities in the different system components. The use of formal specifications for describing the behavior of these components facilitates the systematic analysis of the system and its implementation. So-called formal description techniques (FDT's) have been developed in recent years for the formal description of the communication protocols and services developed in the context of the Open Systems Interconnection (OSI) standards. However, so far they have not been widely used. This paper gives an introduction to this area, presents the description techniques used for writing the OSI protocol standard specifications, and discusses the reasons that limit a wider use of formal specifications in this context.

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 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 many protocols.

A collection of standards of communication protocols and services are being developed for Open Systems Interconnection (OSI) [OSI 83, Knig 88] intended to allow the interworking of heterogeneous computer systems for a variety of applications. In this context, the

* This work is partly supported by the Natural Sciences and Engineering Research Council of Canada.

protocol specifications are of particular importance, since they represent the standards which are the basis for the implementation and testing of compatible OSI systems. The standard specifications are usually written in natural language, augmented with certain limited formalisms.

In addition, formal description techniques (FDT's) have been developed for application to OSI and have been used for the description of certain OSI protocols and services. The formal nature of these specifications make it possible to apply certain automated tools during the protocol development life cycle. After being developed over a period of approximately 8 years, these techniques are now ready for being applied. However, so far they have only found relatively limited use.

After an overview of formal specifications for communication protocols, this paper explains the present trends in relation with the description of OSI standards and discusses certain factors that seem to limit the application of formal methods in this area.

2. Formal specification of communication protocols and services

2.1. The protocol engineering life cycle

Similar to the well-known software life cycle, the development and implementation of communication protocols involves a number of steps and activities listed below. The protocol specification plays a key role in all these activities, and the use of formal specifications allows the partial automation of some of these activities [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. Implementation assessment is a more general term and includes testing of other implementation requirements. 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.

2.2. Overview of specification methods

Descriptions of communication services and protocols must be both easy to understand and precise - goals which often conflict. The use of specifications 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 [Parn 77] apply also to protocols.

Many different formal specification languages have been developed for various purposes, and many of them have been applied to the description of distributed systems and communication protocols. Certain description methods, such as finite state machines (FSM), grammars, Petri nets, process algebras and data types, have been used successfully for the description of certain aspects of the behavior of communication protocols, however, they do not seem suitable for describing all aspects of typical OSI protocols. Other methods, such as high-level programming languages, abstract data types, logic programming (e.g. Prolog), and temporal logic, seem to be capable to describe all aspects of such protocols, however, none of these methods is generally recognized as the best choice for writing formal protocol specifications. Further details can be found in [Boch 89d, Boch 87c, TSE 88] and in the proceedings of the yearly IFIP conferences on "Protocol Specification, Testing and Verification" and "FORTE" (e.g. [Brin 89], [Vuon 90]).

With the beginning work on the standardization for OSI at the end of the seventies, some people recognized that formal specifications could be useful in the development of OSI standards. Special groups discussing "formal description techniques" (FDT) for application to OSI were formed within ISO and CCITT in 1980 [Viss 86, Sara 86]. These groups developed three specification languages, Estelle, LOTOS, and SDL, further described in the next subsection, which are suitable to cover all aspects of communication protocols and services. However, the effective use of these languages in the OSI area, so far, has been relatively slow. This may be partly explained by the competition between these three languages, which each have certain advantages, and by the difficulty many people have in learning a new language.

2.3. Formal description techniques (FDT's) for OSI

In Estelle [Budk 87, Este 89], a specification module is modelled by an extended FSM. The extensions concerning the aspects of interaction parameters and state variables are covered by type definitions, expressions and statements of the Pascal programming language. In addition, certain "Estelle statements" cover aspects related to the creation of the overall system structure consisting in general of a hierarchy of module instances. Communication between modules takes place through the interaction points of the modules which have been interconnected by the parent module. Communication is

asynchronous, that is, an output message is stored in an input queue of the receiving module before it is processed.

SDL [Sara 87, SDL 87] has the longest history; a subset of the present language was already recommended by the CCITT in 1980. It is also based on an extended FSM model. For the aspects of interaction parameters and state variables, it uses the concept of abstract data types with the addition of a notation of program variables and data structures, similar to what is included in Estelle. However, the notation for the latter aspects is not related to Pascal, but to CHILL, the programming language recommended by CCITT. The communication is asynchronous and the destination process of an output message can be identified by various means, including process identifiers or channel names.

LOTOS [Bolo 87, Loto 89] is based on an algebraic calculus for communicating systems (CCS [Miln 80]) which includes the concepts of finite state machines plus parallel processes which communicate through a rendez-vous mechanism which allows the specification of rendezvous between two or more processes. Asynchronous communication can be modelled by introducing queues explicitely as data types. The interactions are associated with gates which can be passed as parameters to other processes participating in the interactions. These gate play a role similar to the interaction points in Estelle. The aspects of interaction parameters and state variables are covered by an algebraic notation for abstract data types, called ACT ONE [Ehri 85], which is quite powerful, but would benefit from the introduction of certain abbreviated notations [Scol 87, Boch 89h] for the description of common data structures.

In contrast to the other FDT's, SDL was developed, right from the beginning, with an orientation towards a graphical representation. The language includes graphical elements for the FSM aspects of a process and the overall structure of a specification. The aspects of interaction parameters and state variables are only represented in the usual linear, program-like form. In addition, a completely program-like form is also defined (called SDL-PR) which is mainly used for the exchange of specifications between different SDL support systems. Presently, there is also a joint work item in ISO and CCITT for the development of a graphical representation of LOTOS.

A comparative evaluation of the three FDT's is difficult to do. The following subjective statements address some of the issues: It seems that Estelle and SDL have the advantage of using well-known concepts of FSM and programming languages which make the intial understanding of the languages easier. The graphics aspects of SDL are also helpful in this respect. On the other hand, LOTOS has relatively few, but powerful language constructs which makes the learning of the complete language easier. LOTOS specifications often tend to be more abstract than specifications written in Estelle or SDL, which are often implementation-oriented. The concepts in the latter languages can be more directly related to typical implementation constructs. For the description of service access points, the rendezvous mechanism of LOTOS is better than the asynchronous message passing of Estelle and SDL, since the latter do not allow a complete specification without including implementation choices [Boch 88h]. The LOTOS syntax seems to be more natural than the FSM-oriented syntax for the description of test cases. The formal definition of Estelle and LOTOS seem to be more readily usable for the

construction of tools than the formal definition of SDL (which is given as an annex of the Recommentation). An attempt of a critical evaluation and comparison of the three languages can be found in [Brui 87].

3. Specification methods for describing OSI standards: present trends

3.1. Use of FDT's

As mentioned earlier, the use of the FDT's has been limited. Certain standardization subcommittees have developed a "guideline" document [FDT GL] which shows how the concepts of the OSI Reference Model can be described by each of the three FDT's. It also includes some tutorial example specifications. Formal specifications in LOTOS have also been developed for the Transport and Session protocols and services (see for instance [Eijk 89]). Many other specifications have been developed outside the standardization groups, largely by research groups experimenting with the use of FDT's.

An ISO/CCITT policy statement concerning formal specifications of OSI protocol standards indicates that it is up to the group defining a protocol standard to decide whether they want to develop a formal specification, and which FDT should be used for this purpose. A formal specification can be included in the standard document (otherwise written in natural language) in the form of an annex, which in some cases may have no standard status, while in other cases it could play the role of the standard reference.

3.2. The "Abstract Syntax Notation One" (ASN.1)

This is a notation for describing data structures [ASN1], 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 [ASN1 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.

3.3. Finite state machines (FSM's)

FSM models are often used for describing protocols. The inputs and outputs of the FSM correspond to the PDU's exchanged with the remote entity and also to the service primitives exchanged with the local user of the protocol entity. Various notations can be used to represent FSM's, such as transition diagrams, regular grammars, or transition tables. OSI protocol standards often include (sometimes as annexes) a transition table or diagram which describes the allowed order of interactions. FSM models are also used to define the order of interactions that are allowed to occur at a given access point of a communication service. It is important to note that these descriptions do NOT define the parameter values of the interactions.

3.4. The "Table-Tree Combined Notation" (TTCN)

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 being 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 developping FDT's for this purpose, which in the author's 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 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 FDT's would be desirable.

3.5. Object-oriented specifications

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.

ASN.1, TTCN, and the notations for objects have in common that they were developed by standardization committees that felt the need for a formalized notation in relation with their main task, namely the development of a particular protocol standard, or sets of standards. While the syntax of these notations is defined in a formal manner, the semantics of the language constructs are described informally, sometimes not very precisely [Kest 89].

3.6. Other developments

Independently from the standardization area, formal protocol specification are sometimes used in the industrial sector. In many cases, the existing formal specification of protocol standards are used for protocol implementations. In other cases, non-standard protocols are developed using a formal specification language. In certain cases, one of the standardized FDT's is used, in other cases certain in-house languages [Schu 80, Holz 88].

In the FDT context, ongoing research centers around the semi-formal specification languages ASN.1 and TTCN, and the FDT's Estelle, LOTOS and SDL. Main issues are the relation between the semi-formal languages with the FDT's, and their use for the formal specification of the protocols and services. Based on experience with the different FDT's, it is also expected that some pragmatic decision on the use within OSI of one or the other FDT will be made.

In the meantime, certain researchers propose improvements to the existing, standardized FDT's. Such proposals include for instance the introduction of rendezvous interactions to Estelle together with certain simplifications concerning parallel processes [Cour 88], the introduction of abbreviated notations for defining common data stuctures, such as enumerations, records and arrays, in LOTOS [Scol 87], and the extension of SDL to handle

non-determinism and separate input queues [Orav 88]. It is not clear what impact these proposals will have on the use of the respective FDT's.

4. Difficulties with using FDT's in practice

As mentioned earlier, the use of FDT's so far has been quite limited. The discussion of this section is intended to provide a better understanding of how successful the use of FDT's has been to date, and what the reasons are that have limited their application.

4.1. How successful are the FDT's ?

It is not clear what is the best measure to determine the success of a specification language, in particular for a specification language intended for describing standards. The following points, each, shed some light on the success of the FDT's described in Section 2.3. The order in the list does not necessarily reflect the importance of these points.

(1) Selection of one FDT for use in the OSI area: The standardization committees have not (yet) agreed to give preference to one of the three FDT's. This complicates their application to OSI standards; the possibility of having to support several FDT's reduces their value.

(2) Application to the description of standards: The number of formal specifications of OSI standards developed by standard committees is very limited. The application of the FDT's to the OSI Application layer protocols (layer 7) presents the problem that ASN.1 is used in these standards and no translation between ASN.1 and the FDT's has been defined within the standardization community, although such translations have been proposed by the research community. Most existing FDT specifications have been developed by research groups, including those groups that were involved in the development of the FDT's. However, these specifications are often incomplete, and have not necessarily been checked for consistency with the (informal) protocol standard.

(3) Use during standard development: The advantage of using an FDT during the design of a new protocol standard has been pointed out by the FDT developers. For the development of the OSI protocols of layers 2 through 6, the FDT's could not be used because when these protocols were developed, the FDT's were not completely defined. Now that they are defined, they are still not much used for the new standards that are presently developed for the OSI Application layer. As mentioned in Section 3, other ad hoc languages are mainly used instead.

(4) Support tools: It has sometimes been argued that a new language is not accepted by the community unless good support tools (e.g. compilers) are available, and that good support tools are only developed once the language has many users, thus justifying the investment in the tool development; a chicken-and-egg problem. Although few professional support tools are now commercially available for FDT's, there are a good number of tools that were developed in the research environment and which are sufficiently well designed and implemented to be suitable for real protocol development projects, in particular for the creation of specifications, their validation and implementation (see [Boch 87c] for a survey).

(5) Application of tools to protocol development: As FDT's were not used in the development of OSI standards, FDT tools for protocol design validation could not be used directly. However, certain FSM-based validation tools were successfully used for debugging the transition tables of several OSI protocol standards, including the Transport and Session. Certain implementation tools based on Estelle (and some of its precursors) were used for obtaining implementations of OSI protocols (including Transport, Session and FTAM) from formal specifications which were developed partly outside the standardization committees.

(6) Stability of language: As mentioned in Section 3.6, certain changes have been proposed to the FDT's. Nevertheless, it seems that the FDT's cover essentially the requirements for communication protocol and service specifications [Brui 87, Boch 90]. However, for use with the OSI Application layer protocols, their relation with the data structures of ASN.1 and the object-oriented concepts used for these protocols must be clarified.

(7) Awareness about language: It seems that the members of OSI standardization committees and many practitioners in the field of communication protocols are aware of the existance of the FDT's, but few are enough familiar with them in order to use them. Many people find them hard to understand. It is to be noted that, in contrast to ASN.1 and the object-oriented notations, the FDT's were developed by standardization subcommittees which had only limited contact with those groups working on protocol standards; this makes their acceptance more difficult.

Overall, the above points seem to indicate that the FDT's are not well accepted. It is hoped that further experience with the use of FDT's will make them more acceptable.

4.2. Factors influencing the use of formal methods

In view of promoting their use, it is important to understand what the reasons for the low acceptance are. 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.

(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 FDT's 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 the 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/hardward 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.

The following factors, specific to the area of OSI standardization, also seem to have a impact on the acceptance of FDT's:

(7) Economic issues: Certain economic considerations discourage formal specifications. For instance, certain standard specifications contain intended ambiguities, which would be difficult to model formally. The existance of recognized formal specifications would also reduce the required implementation effort, thus increasing competition.

(8) Time frame: If the formal specification of a protocol is developed after the standard, and not at the same time, most implementation efforts (which occur when the standard appears) can not take advantage of the formal specification; it comes too late.

(9) Test cases versus specification: There seems to be a certain trend in OSI to consider standardized test cases (with associated verdicts) as a substitute for a non-ambiguous specification.

5. Conclusions

In software engineering and in the development of distributed systems, there seems to be a general trend towards a formalization during the requirements and design phases of system development. In the area of OSI, various formalisms with limited scope (e.g. FSM models, ASN.1 data structure definitions) are used for making the specifications of protocol standards more precise. In addition, three so-called Formal Description Techniques (FDT's) were developed, namely Estelle, LOTOS, and SDL, which can be used to provide complete specifications of communication protocols.

After arousing initially much expectations during their development, FDT's are presently not well accepted in the work of the standardization committees, and in the wider context

of industrial protocol development. A variety of factors, discussed above, can be put forward to explain this situation. It seems that a larger acceptance of formal description methods can only be obtained if these factors are considered. In summary, this means that adequate user training should be provided, the concepts of the specification language should be simple, the relation between the informal and formal specifications should be easily made by the reader, the language should have wide applicability, the support tools should be simple to use, and should be integrated to the general software or hardware development environment.

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).

[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.

[Boch 88h] Gregor v. Bochmann, Alain Finkel, "Impact of queued interaction on protocol specification and verification", Proc. Intern. Symp. Interoperable Inf. Systems (ISIIS), Nov. 1988, Tokyo, pp. 371-382.

[Boch 89d] G.v.Bochmann, "Protocol specification for OSI", to be published in Computer Networks and ISDN Systems.

[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.

[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.

[Bolo 87] T.Bolognesi and E.Brinksma, "Introduction to the ISO Specification Language Lotos", Computer Networks and ISDN Systems, vol. 14, no. 1, pp.3, 1987.

[Brin 89] Protocol Specification, Testing and Verification IX, E.Brinksma et al. (eds.), North Holland Publ., 1989.

[Brui 87] The SPEC Consortium and J.Bruijning, "Evaluation and integration of specification languages", Computer Networks and ISDN Systems 13 (1987), pp. 75-89.

[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.25, 1987.

[Cour 88] J.P.Courtiat, "Estelle*: a powerful dialect of Estelle for OSI protocol description", Proc. IFIP Symposium on Prot. Spec., Verif. and Testing, Atlantic City, 1988.

[Ehri 85] H.Ehrig and B.Mahr, Fundamentals of Algebraic Specifications 1, Springer Verlag, 1985.

[Eijk 89] P.H.J. van Eijk, et al., "The formal description technique LOTOS", North Holland Publ. 1989.

[Este 89] ISO IS9074 (1989) "Estelle: A formal description technique based on an extended state transition model".

[FDT GL] ISO Tech. Report, Guidelines for the use of formal description techniques for OSI specifications, 1989.

[Holz 88] G.J. Holzmann, "An Improved Protocol Reachability Analysis Technique", Software-Practice and Experience, Vol. 18 No. 2, February 1988, pp. 137-161.

[Kest 89] S.Kesti and K.Ronka, "Use and applicability of ASN.1", Proc. 2-nd Int. Workshop on Protocol Test Systems, Berlin, Oct. 1989 (North Holland Publ.).

[Knig 88] K.G.Knightson, T.Knowles and J.Larmouth, Standards for Open Systems Interconnection, McGraw-Hill, 1988.

[Loto 89] ISO IS8807 (1989), "LOTOS: a formal description technique".

[Miln 80] R.Milner, "A calculus of communicating systems", Lecture Notes in CS, No. 92, Springer Verlag, 1980.

[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.

[Orav 88] F.Orava, "Formal semantics of SDL specifications", Proc. IFIP Symp. on Protocol Specification, Testing and Verification VIII, North Holland Publ. 1988.

[Parn 77] D. Parnas, "The use of precise specifications in the development of software", in Proc. IFIP Congress 1977, pp.861-867.

[SDL 87] CCITT SG XI, Recommendation Z.100 (1987)

[Sara 86] R.Sarraco, Proc. IFIP Congress 1986, Dublin.

[Sara 87] R.Sarraco and P.A.J.Tilanus, "CCITT SDL: Overview of the language and its applications", Computer Network and ISDN Systems, 13 (1987), pp. 65-74.

[Schu 80] G.D.Schultz, D.B.Rose, C.H.West, and J.P.Gray, "Executable description and validation of SNA", IEEE Trans. COM-28, no.4 (April 1980), pp.661-677.

[Scol 87] ISO 97/21 N1540, "Potential enhancements to Lotos", 1986.

[TSE 88] "Special Issue on Tools for Computer Communication Systems", IEEE Tr. Software Engineering, Vol. 14, March 1988.

[Viss 86] C.Vissers, "Formal description techniques for OSI", Proc. IFIP Congress 1986, Dublin.

[Vuon 90] Formal Description Techniques (FORTE '89), S.Vuong (ed.), North Holland Publ., 1990.