Formal Description Techniques for Distributed Computing Systems, the challenges for the 1990's.

Chris A. Vissers[1] Jeroen van de Lagemaat[2] Luis Ferreira Pires

University of Twente, Fac. Informatics
7500 AE Enschede, NL

Abstract

Initially FDTs where developed within ISO and CCITT for specification, at a high-level of abstraction, of distributed systems. Research is now being performed on the use of FDTs to support the complete implementation trajectory. In this paper we discuss a number of such research activities that are conducted within the framework of the Lotosphere project(*). The paper discusses aspects of design methodology, correctness preserving transformation, the reflection of design criteria, the role of pre-defined specification and implementation constructs, and formal approaches to conformance testing. Furthermore some insight is given in the development of a comprehensive tool-set that supports these aspects of design methodology. The paper concludes with some experience obtained from the application of these methods and tools to some realistic pilot implementations: an ISDN and MHS application and a Transaction Processing application.

1. Introduction

In the past decade three Formal Description Techniques (FDTs) have been developed in the framework of the international standardization organizations ISO and CCITT, LOTOS [1, 2, 3], Estelle [4], and SDL [5]. The development was predominantly guided by criteria that characterize the requirements capturing phase of the distributed systems life cycle: unambiguous, comprehensiveness, and clarity. These criteria naturally concur with the realm of international standardization: international standards and recommendations only have to be specified. Moreover, these specifications are constrained by the dominant requirement of implementation independence. Any attempt towards more implementation oriented is considered to be in direct conflict with the widely adopted principle of openness of international standards.

Despite of numerous difficulties in the development of these FDTs [6], these FDTs are now published as international standards, and applied at a world scale, and an increasing amount of formal specifications of both standards [7 - 14] and of propriety systems are becoming available. These specifications have to serve as the authoritative reference for product implementations and product testing. Naturally the question is presenting itself how the FDTs can be used both to support the complete design trajectory, including implementation and product testing, and to effectively represent the design concepts of the complete design trajectory.

Recent research activities are aimed at addressing this challenge, e.g. within the ESPRIT and RACE programs the CEC supports several programs such as SEDOS [15], PANGLOSS and SPECS. Currently the ESPRIT II project 2304, LOTOSPHERE is undertaken, aiming at the development of industrially applicable methods and tools based on LOTOS, and demonstrating the applicability of methods and tools for the design and implementation of both ISDN and ISO applications. Recently accepted projects are also focussing on the industrial applicability of design methods for specific fields of application e.g. COMPLEMENT (ESPRIT II 5409) on real-time systems .

This paper is organized as follows: First an introduction to design methods in general is given together with a discussion of the typical characteristics that make FDTs extremely suitable as key languages for the methods. Second different activities within LOTOSPHERE will be discussed where the FDT LOTOS is used. Finally concluding remarks will be given on the industrial applicability of methods and tools developed within LOTOSPHERE, based on actual experience in applying these to realistic sized industrial products.

2. Design methodology and its support by FDTs.

Information Technology industries will depend more and more on methods to control the development of complex systems. Both quality issues and cost considerations will determine the ability of industry to play an important role in the IT market. In this section we present the basic concepts of design, the requirements on methods used during the design process in order to meet the industrial constraints, and the role FDTs play in this context.

(*)The authors act as project manager[1] and project co-ordinator[2] of the ESPRIT Lotosphere project (2304) respectively. This project is using LOTOS as a basis for a formal approach to design, implementation and testing.

2.1 Design

The process of defining and building systems can be characterized as follows:

- A design process is the activity in which user requirements are formulated and transformed into a real system;
- A design methodology is a set of methods that can be used to formulate the user requirements and transform them into a real system;
- A design trajectory is a sequence of design steps, produced by step-wise modification (e.g. refinement), that starts with the formulation of the user requirements and terminates with the production of a concrete instance of the desired system.

A design methodology will guide the developer(s) of a system. It should be capable of both reflecting the requirements imposed on the system and of applying different design criteria in a structured way.

Generally a design trajectory is viewed as a top-down process, leading towards a real system i.e. degrading the level of abstraction and increasing the number of system properties taken into account, as illustrated in figure 1.

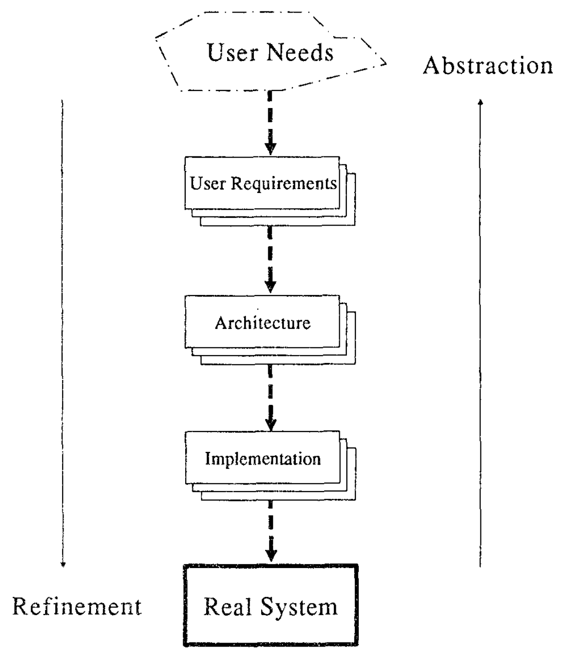

figure 1, Simplified view of design trajectory

The basic element of design is the design step. In each step a design of a higher level of abstraction is transformed into a design of a lower level of abstraction. The lower level design(s) shall be "correct" w.r.t. the higher level design. By ensuring this consistency in design a separation of work and responsibilities can be made effective. Each design step is guided by design knowledge, methods and criteria relative to its place in the total trajectory. Each step can be characterized by its source and target description of the system and the design criteria taken into account.

Refinements of design process Several possibilities of refinement of this view may be considered [16].

One can take an cyclic approach and increment the requirements taken into account each cycle. In the first cycle a limited set of user requirements are taken through the design trajectory. In a next cycle additional requirements are added until all requirements are satisfied. This approach has several advantages. First, it allows to use in each next cycle the experience (in applying methods) obtained in previous cycles, whereas this experience comprises not only top-down, but also bottom-up, experience. Second, early design cycles do not need to consider the burden of all design details. Thus they can be exercised quickly providing insight in the design problem accordingly. The cyclic approach allows also the parallel operation of different expertise groups like architects and implementors.

The design can also be considered as a search through different alternatives at every stage of the design process. The design trajectory can than be considered as a tree-pruning process, as illustrated in figure 2.

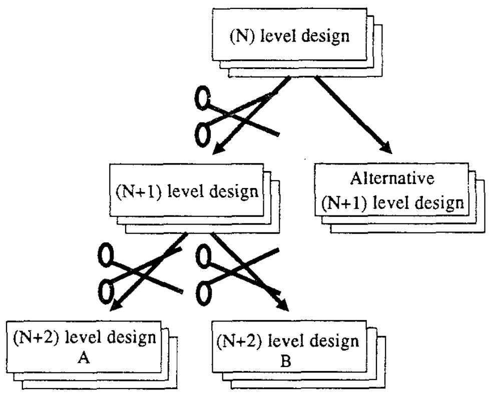

figure 2, Design as a tree-pruning process

2.2 Role of FDTs in Design

The characteristics of the complete design process given above lead to a set of requirements for methods used as part of it, in order to improve the quality, speed and control of design and to reduce the resources needed:

- Description or representation methods used at each level of the design shall be capable of reflecting all properties of the system taken into account so far, and provide a means to analyse the description w.r.t. these properties. Therefore a representation method is only of practical use if it is based on the syntactical and semantical rules of a design language. This requirement is induced by the fact that there should be a consistent relationship between the properties that are expressed by a design and the language constructs expressing them. The semantical rules form a basis for analysis.
- Methods used to perform design steps shall provide means to achieve the desired level of abstraction of the target description while taking into account the applicable design criteria.
- Each method used must be comprehensive and concise. Nevertheless these methods are applied to very complex systems. Tools are therefore a pre-requisite and shall relieve the designer(s) of checking the syntax or static semantics, ensure consistency of design, guide in selection out of possible alternatives, etc.

Design Language A language is called a broad spectrum design language if it can express designs at many different abstraction levels along the design trajectory. Such a design language necessarily needs to be general purpose; it is impractical to reflect each different design property by a special purpose language element. The language must also allow the representation of the intended properties directly and clearly. This imposes high demands on the expressive power of the design language and its model. In many cases not all properties of a design can be expressed in the model used. Limitations in the expressive power of a model may force the designer to use multiple models and to move between models in the course of the design process to ensure that all relevant properties of the real system are covered in the specifications at each level of abstraction

By using a single design language the designers can improve their communication. This is very important if different designers perform different design steps.

The consistency of properties at successive levels of design is also facilitated if a single design language is used. Using a single design language may also avoid re-writing of parts of a design that are not affected by the design decisions taken in a specific design step.

FDTs as design languages FDTs, languages which are based on a formal model, have a high potential to support the design process and enhance its efficiency and quality. The underlying formal model enables a rigid and unambiguous expression of designs, that subsequent can be analysed and verified for consistency by using mathematical methods. The formality of the model also allows the development of software tools that can support the design process and automate parts of it.

The experiences gained with the application of FDTs so far show that they can be used as broad spectrum design languages very effectively [17, 18, 19]. Tools have been developed for several FDTs which enable analysis of descriptions.

Uncertainties w.r.t. the use of FDTs as a design language are:

- Probably not all properties of systems can be expressed, e.g.in general real systems can not be specified using an FDT, and
- It may not be feasible and/or practicalnot to base all methods and tools necessary within the design trajectory on the model underlying an FDT.

The real challenges are to investigate these limitations and to overcome the limitations as much as possible by a combination of research and practical/realistic application.

3. LOTOS and Design

In April 1989 the ESPRIT II project 2304, LOTOSPHERE, started with a consortium of 17 partners from 7 European countries. The central theme of the LOTOSPHERE project is the conversion of LOTOS into a viable, fully tool supported, system design and development methodology. Some specific project objectives and advances are:

- The development of a methodology and formal models for system design and development, incorporating design structuring techniques, correctness preserving transformation criteria and techniques, testing, and maintenance of LOTOS.
- The production of a consistent and coherent industrial applicable integrated LOTOS development environment, resulting in an integrated tool-set which supports the complete design and development trajectory and which is equipped with an effective user-interface.
- The preparation of large-scale specifications in LOTOS of OSI and ISDN products, in liaison with standardisation bodies where appropriate, realisation of pilot implementations of realistic sized products based on these specifications using the methodology and tools developed and promotion of the strategic uptake of the project methodology and tools by industry

Furthermore the project aims at advancing the European IT industries by contributing to a formally-based approach to software engineering based on promising new developments such as architectural design structuring, demonstrated for a broad range of applications. It promotes European cooperation especially between researchers, manufacturers and users and contributes towards standardisation in the areas of OSI, ODP and ISDN.

Within the project results of previous research related to LOTOS, and experience of industrial partners on realisation of complex systems will be used.

In Lotosphere the design methodology is based as much as possible on a single broad spectrum design language with a formal model. For this language we have chosen the FDT LOTOS because it has a high expressive power, allowing not only to make unambiguous, but also comprehensible, and concise specifications [18, 19]. Its formal basis makes it a prominent candidate to achieve the above-mentioned goals [2, 20].

An important industrial criterion is also that LOTOS is internationally standardized (IS 8807), implying that its definition is stable. Standardization also promotes the increasing availability of advanced design support tools.

The project is divided into three workpackages : Methodology, Tools and Applications, according to the objectives given above. Each workpackage is divided in tasks which cover parts of the work. In the following sections we discuss the research activities of the first two workpackages. At the end of this paper we present some results obtained in the third workpackage.

3.1 The LOTOSPHERE Method

Research on a LOTOS based design method is divided into four separate activities:
- Design Structuring Techniques
- Correctness preserving Transformations
- Testing
- Language Enhancements

Design Structuring Techniques It is considered that automatic design is difficult due to both the complexity of designs and the necessity to add design decisions during the design process which can only be taken by humans and which do improve the quality of the final realised system, e.g. on performance or modularity.

Key elements of a design trajectory are the design requirements and design decisions, called design criteria. The central problem is how to reflect and structure knowledge, methods and the design criteria in the different design steps using FDTs.

The following aproaches are used:

- Pragmatic guidance for performing design steps will be found in identifying for every step the level of abstraction of the resulting system description. This guidance dictates the kind of design decisions that must be considered in each design step and their relative position in the design trajectory. It also dictates the transformation strategy to be applied, and should work as a binding element among the other concerns. Relative evaluation of possible distinct design decisions for conformance to non-formalized properties,i.e. properties that are not expressed in the syntax and semantics of behaviour expressions and abstract data types of LOTOS, shall be covered by this method, providing the means to designers to proceed in the design trajectory.

- Specification styles specific for different levels in the design process are identified which are capable of reflecting the relevant properties according the qualitative design criteria. The same language constructs are used at all levels which facilitates a systematic and general purpose representation of properties and criteria. High quality specifications can be obtained which can be transformed systematically into equally high quality implementations.

- In most cases the user requirements will be described in natural language, although part of the user requirements can and will be expressed formally. The formulation of the architecture will be the first occasion where LOTOS can be fully used. Experience indicates that many properties of architectures can be conveniently expressed in LOTOS at the appropriate level of abstraction. However, significant properties of architectures cannot be formalized at architectural level. Examples are requirements for performance, absolute time, cost, or requirements that can only be globally indicated, such as "suitable for stream oriented traffic", "fault tolerant", or "robust".

- In most cases it is not possible to describe real systems using LOTOS nor is it feasible to establish a formal relationship between LOTOS and implementation languages such as Pascal, C, or hardware design languages such as HDL, RTL, FSMs etc. The approach is taken to define pre-defined implementation constructs which are pairs of implementation oriented LOTOS descriptions and implementation constructs expressed in an implementation language, such as Pascal, C etc. For each pair it is assured that the requirements imposed by the LOTOS description are met by the implementation construct. The pre-defined implementation construct are used as building bricks for the lowest level LOTOS specification, called implementation, and serve as a guide for the design trajectory.

- The application of design requirements and decisions which reflect general design principles, also called qualitative design criteria, such as orthogonality, propriety, generality, open endedness, symmetry, parsimony, robustness, performance, etc. are presented by means of examples.

- In order to achieve consistency in design the following alternatives are considered to ensure that consistency is not violated in (a sequence of) design steps: Correctness preserving Transformations, Verification, and Testing. All three are based on the semantics of the language used, in particular on the equivalence relations which hold for specifications, also referred to as second order semantics.

Correctness preserving Transformations The following requirements apply to transformations when used as a design step:

- Each transformation must fulfil one or more design criteria
- The resulting specifications must be "correct" w.r.t. to the source specification, and
- Redundancy must be avoided.

The following transformation paradigms are identified:

- (De-)composition of atomicity. Atomicity means that something either happens completely or not at all. An atomic action, like an event in LOTOS, can be decomposed into several actions with the requirement that the result of these actions is equivalent to the atomic action. Different (de)compositions of interaction, interaction point and functionality are envisaged.
- Transformation of datatypes into either compatible data structures and/or into process descriptions.
- Transformation into pre-defined implementation constructs.

To obtain these transformations the following approaches are used:

- Non-standard semantical interpretations, called views of specifications, are defined to express the design criteria taken into account by a transformation which can not be expressed by the standard semantics of LOTOS, such as distribution of functional behaviour over modules, robustness, ergonomics of user-interface, costs, etc..
- Correctness of transformations is preserved by ensuring that equivalence relations defined for LOTOS hold.

The results obtained so far are:

- For each transformation the requirements are identified
- Several transformation problems have been formalized and some of them have been resolved.
- A rewrite system is defined for LOTOS processes which preserves observational equivalence.
- A temporal logics semantics for Basic LOTOS has been defined and will be used to deduce partial properties of LOTOS specifications.
- Definitions of executability, efficiency and style w.r.t. LOTOS data type specifications have been provided. Concepts of data type transformation are introduced and exemplified.
- A preliminary implementation oriented model for LOTOS, called LOTOMATION has been defined.

Testing Transformations can not always ensure correctness preservation throughout the complete design process and complementary approaches are needed to ensure consistency in desi n. For the fast majority of practical applications of design methodologies it is not feasible to verify on a formal basis whether a lower level design is correct w.r.t. to a higher level design. Even if the problem can be solved in theory, practical limitations such as time and space will prohibit this. Testing can be a viable alternative.

Within the design trajectory testing can play a role at several stages. Most widely accepted is the use of Conformance Testing as standardised by ISO and CCITT [21]. Furthermore testing can be applied also to intermediate designs which facilitates early detection of errors. Also testing can be applied to parts of design, reducing the resources needed for final integration of parts of a system. Finally white box testing can be used.

For testing to be of any use it is vital that high quality test sequences are available. This means that tests shall be correct w.r.t. the specification which forms the basis for testing, and that adequate coverage, i.e. optimized w.r.t. functionality tested and costs, is obtained.

Within the LOTOSPHERE project test derivation theory [20] will be applied to define algorithms to derive test sequences for equivalence relations. The resulting tests can be applied both to intermediate LOTOS specifications and to final realisations. Also the representation of tess, selection of tests and design of test systems will be addressed. Contributions will be made the ISO on this item.

Language Enhancements Language modifications are kept to a minimal. A wide-spread industrial use of LOTOS is only feasible if its definition remains stable and rapid introduction of new concepts generally turns out to be rather counter-productive.

Based on the experiences in both the definition of a complete design methodology, definition and implementation tools, and in applying the methodology and tools in practise, modifications are proposed to the LOTOS language. These modifications include:

- Introduction of modules in LOTOS, to increase tool performance and improve separation of work among different designers.
- Introduction of time in LOTOS.
- Refinement of event structures.
- Modification/extension of the data type part

Work on these items has just been started and is expected to provide results very fast in order to use these results in the remainder of the project.

Some contribution have been made to the definition of a graphical version of LOTOS which will be standardised as part of IS 8807.

3.2 Tools.

Within the design process tools are vital. Designers should not be experts in the theory of the models underlying methods used in design, but should focus on the design itself. Therefore tools must incorporate the methods and provide the user practical manipulation facilities. Furthermore the inherent complexity of most design demands tool support.

As LOTOS is an abstract technique, the existing techniques for tool production for traditional software and hardware design can not be used. Tool development will be based among others on attributed grammars and term rewrite systems.

The requirements for the development of tools are:

- The tools shall form a coherent set which supports the complete LOTOSPHERE design methodology.
- The tools must be equipped with a powerful and attractive user-interface.
- The tools must be efficient.

In order to obtain a LOTOSPHERE Integrated Tool Environment (lite) the following actions are undertaken:

- Existing meta-tool environments which can be used for the production of tools are evaluated.
- One internal representation within the tool-set, called Common Representation, has been defined in order to insure consistency and increase efficiency.
- An overall Architectural Design has been produced.

A first version of lite is used within the project which incorporates syntax and static semantics checkers, a LOTOS editor, a simulator, compilers for data and processes and a tools to integrate comments into LOTOS specifications. The tool-set is equipped with an X-windows user-interface.

In future tools will be integrated to support further analysis of specifications (report generators, cross reference generators), data manipulation, verification (e.g. persistency checker), and tools to support transformations and test derivation/selection.
User manuals for tools will be provided.

Software development is be undertaken according to the ESA software engineering standard.

4. What about the challenge?

Within LOTOSPHERE one workpackage is devoted to using the LOTOSPHERE design method and tool-set to produce industrial realisations of:

- An ISDN application Mini-mail. This application provides the user with a limited message handling facility for telephone in case a called partner is absent or busy. For this purpose the telephone set is equipped with a keyboard and a small display
- ISDN layer 3
- A Transaction Processing application

Until recently work has been done on the production of initial specifications according to the styles recommended in the method. Tools have been used for writing, checking and analysing these specifications.

The experiences gained include:

- The structuring principles are very suitable for the initial phase of the design trajectory,
- Specification of a new application, Mini-mail, while defining the application itself, showed to be efficient. A layered approach is used to separate user-interface, message manipulation and message transport.
- Specification of an ISO standard which is still evolving causes a lot of problems. Relevant experiences and improvements to the standards are forwarded to ISO.
- Additional practical guidance is needed on the application of the method in the subsequent phase of the design trajectory.
- The performance of the pre-project tools which where used have inadequate performance both in time and computer memory.

Target machines on which the applications will be realized are identified; for TP a combination of Unix, C-language and the OSI Development Environment ISODE, for Mini-mail various machines like telephone set, PC, workstation will be used. This selection will form the basis for the definition of the pre-defined implementation constructs for these applications.

For the second project year the appealing part of the challenge will be addressed, i.e. how far do the methods and tools based on LOTOS reach in an industrial environment.

5.Acknowledgements

This work has been partly supported by the CEC under the contract of the ESPRIT II project 2304, LOTOSPHERE

6. References

[1] ISO-IEC/JTC1/SC21/WG1/FDT/C, 'LOTOS, A Formal Description Technique based on the Temporal Ordering of Observational Behaviour', ISO International Standard IS8807, February 1989.

[2] H. Brinksma, 'On the Design of Extended LOTOS, A Specification Language for Open Distributed Systems', PhD Thesis, Univ. of Twente, November 1988.

[3] P.H.J. van Eijk, C.A. Vissers, M. Diaz, 'The Formal Description Technique LOTOS, Results of the ESPRIT/SEDOS project', North Holland 1989.

[4] ISO-IEC/JTC1/SC21/WG1/FDT/B, 'Estelle, A Formal Description Technique based on an extended state transition model', ISO International Standard IS9074, July 1989.

[5] CCITT/SGX/WP3-1, 'SDL, Specification and Description Language', CCITT Recommendations Z.100-Z.104, 1988.

[6] C.A. Vissers, 'FDTs for Open Distributed Systems, a Retrospective and a Prospective View', to appear in Proc. IFIP WG6.1 Symposium on Protocol Specification, Testing and Verification X, Ottowa, Ontario Canada, June 1990 (North Holland).

[7] ISO/TC97/SC6/WG2, 'Protocol for providing the Connection-less Mode Network Service', Draft Addendum 2 Formal Description of ISO 8473', ISO/TC97/SC6/N3938, January 1986.

[8] ISO/TC97/SC6/WG4/Ad Hoc Group, 'Formal Description of ISO 8073 (the Transport Protocol Specification) in Estelle', ISO/TC97/SC6/WG4/N123, March 1986.

[9] ISO/IEC JTC1/SC6/WG4/Ad Hoc Group, 'Formal Description of IS 8073 (Transport Protocol) in LOTOS', ISO/IEC JTC1/SC6/WG4/N233, August 1988.

[10] ISO/TC97/SC6/WG4/Ad Hoc Group, 'Formal Description of the Transport Service in Estelle', ISO/TC97/SC6/WG4/N53, November 1985.

[11] National Bureau of Standards, 'Formal Description in Estelle of the Multi-vendor File Transfer Protocol', American National Standards Institute, Working document number X3T5.5/85-131, 1985.

[12] ISO/IEC-JTC1/SC21/WG6/Ad Hoc Group, 'Information Technology - Open Systems Interconnection - LOTOS Description of the Session Service', ISO/IEC/TR9571:1989(E), September 1989.

[13] ISO/IEC-JTC1/SC21/WG6/Ad Hoc Group, 'Information technology - Open Systems Interconnection - LOTOS Description of the Session Protocol', ISO/IEC/TR9572:1989(E), September 1989.

[14] ISO/TC97/SC6/WG4/Ad Hoc Group, 'Revised text of ISO/PDTR 10023 'Formal Description of IS 8072 (Transport Service) in LOTOS', ISO/TC97/SC6/N5533, June 1989.

[15] M. Diaz, C.A. Vissers, 'SEDOS, Designing Open Distributed Systems', IEEE Software, Vol.6 Nr.6, November 1989, pp24-33.

[16] K. Bogaards, 'A Methodology for the Architectural Design of Open Distributed Systems', PhD Thesis, University of Twente, June 1990.

[17] G. Scollo, C.A. Vissers, A. di Stefano, 'LOTOS in Practice', Proc. IFIP 86, 10th World Congress, Dublin, Sept. 1986, pp 869-875.

[18] ISO-IEC/JTC1/SC21/WG1/FDT/A: 'Guidelines for the application of Estelle, LOTOS, and SDL', ISO DTR10167, January 1990.

[19] C.A. Vissers, G. Scollo, and M. v. Sinderen, 'Architecture and Specification Style in Formal Descriptions of Distributed Systems', Proc. IFIP WG6.1 Symposium on Protocol Specification, Testing and Verification VIII, Atlantic City, USA, June 1988 (North Holland, 1989) pp189-204.

[20] H. Brinksma, R. Alderden, R. Langerak, J. v.d. Lagemaat, J. Tretmans, 'A Formal Approach to Conformance Testing', Proc. of Second International Workshop on Protocol Test Systems, Berlin, Oktober 1989, (North Holland, 1990).

[21] ISO/IEC-JTC1/SC21/ad hoc Group, 'Information Technology - Open Systems Interconnection - OSI Conformance Testing Methodology and Framework', ISO/IEC/DIS 9646.