Towards the new Test Specification and Implementation Language 'TelCom TSL'

Thomas Walter[a] and Jens Grabowski[b]

[a]Eidgenössische Technische Hochschule Zürich, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich/Gebäude ETZ, 8092 Zürich, Schweiz, e-mail: walter@tik.ethz.ch

[b]Universität Bern, Institut für Informatik und angewandte Mathematik, Neubrückstrasse 10, 3012 Bern, Schweiz, e-mail: grabowsk@iam.unibe.ch

Abstract

The development of multi-media and real-time applications requires the assessment of new functional and non-functional properties by conformance tests. The Tree and Tabular Combined Notation (TTCN) is not adequate for this purpose since test cases cannot be specified for most of the new requirements. For this reason, in 1994 we started to develop *TelCom TSL*[1], a new test specification and implementation language for advanced telecommunications applications. TelCom TSL provides facilities for the specification of tests for non-functional properties like timing constraints, quality of service (QoS) aspects, or synchronization of multi-media data streams. In this paper we describe the requirements for QoS testing and present some ideas for the design of TelCom TSL.

1 Introduction and Motivation

For the past decade research and developments in conformance testing and test specification languages have been focussed on TTCN [13] and, but only recently, on concurrent TTCN [14]. TTCN is a test notation which is standardized for the purpose of conformance testing [11]. Conformance testing as understood by ISO and ITU-T is *functional black-box testing* of OSI protocol implementations, i.e., an *implementation under test* (IUT) is meant to be a black box and its observable behavior is compared with the observable behavior of a protocol specification. Conformance testing is concerned with *traditional* protocols and protocol implementations. These protocols and protocol implementations handle one data stream only and, to a limited extent, impose stringent timing constraints. With the upcoming deployment of multi-media applications, like video-conferencing, multi-media archiving and retrieval, or tele-teaching, not only *functional* properties of an implementation are to be assessed during conformance testing but also *non-functional* properties, e.g., timing constraints (as in real-time applications), quality-of-service (QoS) aspects, synchronization of different data streams (audio, video, textual information), should also be taken into account.

To close this gap, in 1994 the University of Berne and the ETH Zürich started a cooperation with the goal to define and implement TelCom TSL, a new telecommunications test specification and implementation language. Our cooperation comprises the following research tasks:

[1]TelCom TSL is an abbreviation for *'TELeCOMunications Test Specification and implementation Language'*.

1. *Requirements analysis:* The requirements that are to be met by the new test case specification language will be analyzed.

2. *Language definition:* The syntax and semantics of the language will be defined.

3. *Validation and simulation:* Appropriate theories for the validation of test cases with respect to functional and non-functional properties will be defined (or adapted if existing approaches are sufficient).

4. *Tools:* We intend to provide tools supporting every activity in the test life cycle.

5. *Implementation:* The test case implementation will be supported by a compiler and runtime libraries.

TelCom TSL shall be general enough to be used for testing *traditional* protocols and new multi-media applications. Currently we are working on the requirements analysis (Task 1) and the language definition (Task 2). In this paper we focus on requirements for QoS testing and present some ideas for the design of TelCom TSL.

2 QoS semantics

Quality-of-Service (QoS) refers to a set of parameters that characterize a connection between communication entities across a network. Typical QoS parameters are [2, 3]: throughput, delay, jitter (*performance* related parameters), or residual error rates, connection establishment failure probability (*reliability* related properties), or presentation coding and security requirements (*miscellaneous* properties).

The negotiation of QoS parameters takes place between calling and called service users and service provider. The QoS semantics define the way how QoS parameter values are negotiated and handled during a connection. We distinguish between *best effort*, *guaranteed*, *compulsory*, *threshold*, and *mixed compulsory and threshold* QoS values.

- **Best effort QoS values.** In this scenario, the calling user requests QoS values that are considered as suggested values, i.e., the service provider has the freedom of lowering the requested QoS value. Similarly, the called service user may also further weaken the QoS value. At the end of QoS negotiation all partners involved have the same QoS values. But this does not imply that the service provider has any obligation for taking preconditions in order to assure that the QoS is maintained for the lifetime of the connection. If the QoS becomes worse the service provider is not even expected to indicate this to the service users. Particularly, no monitoring of the negotiated QoS values is required.

- **Guaranteed QoS values.** In this QoS semantics, the calling user requests a QoS value which is to be regarded as a minimal acceptable value. The service provider has the possibility to strengthen the value or to reject the request if it cannot provide the degree of QoS requested. However, if the request is accepted and the connection is established then the service provider has the obligation for maintaining the agreed QoS values for the lifetime of the connection. In order to achieve this guarantee the

permanent availability of resources allocated to the connection is required. This may imply that further connection requests are rejected since the newly requested QoS values may interfere with the QoS values of already established connections.

One can think of levels of QoS support in between *best-effort* and *guaranteed* QoS. These are *compulsory* and *threshold* QoS semantics [3, 4].

- **Compulsory QoS values.** The value for a QoS parameter to be negotiated may only be strengthened by the service provider and the called service user. When the service provider analyzes a request then the service provider may decide to reject the service request since available network resources are not sufficient to satisfy the desired QoS. However, in the case that the connection is established the QoS of the connection has to be monitored. If the service provider detects that the QoS is not longer provided as agreed during QoS negotiation, then the connection is aborted.

- **Threshold QoS values.** The negotiation of a threshold QoS value follows the same procedure as for a compulsory QoS value. Particularly, any modification to a QoS value is only allowed if it strengthen the QoS value. However, if the QoS value cannot be supported by the service provider then the calling service user gets a feedback on the weakening of the QoS value. Furthermore, whenever the service provider detects a violation of the negotiated QoS value (by monitoring the QoS values of the connection) the service users are informed about this degradation of QoS but the connection is not aborted.

- **Mixed threshold and compulsory QoS values.** A quite interesting possibility is the combination of a compulsory and a threshold QoS value. In such a case, the threshold QoS value must be stronger than the compulsory QoS value. If a connection is established then the negotiated QoS value is greater than or equal to the threshold QoS value. So, if during data transfer the monitored QoS value degrades then first the threshold QoS value is violated which results in a feedback to the service users that QoS of the connections becomes worse and possibly a connection abort may be experienced in the future.

Guaranteed QoS implies the highest degree of commitment for a service provider of maintaining the QoS of a connection. Particularly, the service provider has to take any necessary precautions that under any conceivable circumstances the negotiated QoS values are supported. For threshold and compulsory QoS the obligation of the service provider is to monitor the QoS values and to inform the service users as soon as a violation of the negotiated QoS values is detected (threshold QoS values) or to abort the connection in the case that the QoS has become worse than the negotiated ones (compulsory QoS values).

3 QoS testing issues

From the previous discussion we conclude that different QoS semantics have different impacts of QoS testing. We do not consider the negotiation of QoS values since negotiation of QoS values is a functional property of a protocol specification which can be tested using methods developed for OSI conformance testing [11]. Furthermore, we exclude best-effort

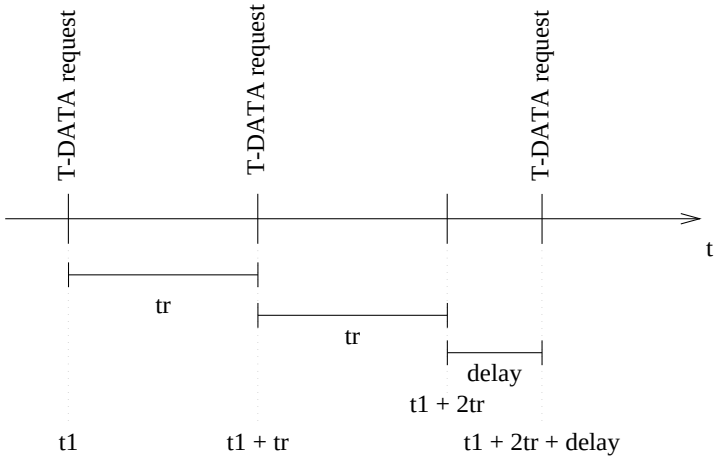

Figure 1: Throughput testing time constraints

QoS semantics from our consideration since no particular constraints on the behavior of a service provider are imposed by this semantics. We argue that OSI conformance testing suffice in this case.

A QoS testing issue is the problem of maintaining QoS values. Threshold, compulsory and guaranteed QoS semantics all require that a service provider, besides implementing the usual protocol functions, is requested to implement additional functions for QoS maintenance, e.g., a monitoring facility. QoS testing, to our understanding, refers to assessing the behavior of a protocol implementation performing QoS monitoring functions. It is not necessary to control and observe the behavior of the monitoring component of an implementation directly, but implicitly by triggering the QoS semantics defined behavior when varying the actual values of QoS parameters. For instance, in case of a threshold QoS value a degradation of a negotiated QoS value should eventually result in an indication of the service users by the service provider that the negotiated QoS value is not further supported. Similar scenarios apply in all other QoS semantics definitions. One can easily imagine that QoS testing imply stringent timing constraints on the behavior of a test system.

As an example we consider a compulsory QoS semantics for the QoS parameter *throughput*. We assume that we test a transport service provider. QoS parameter throughput is defined "*as the ratio of the size of a submitted TSDU to the time elapsed until the occurrence of the next* `T-DATA request` *on the same transport connection*" [3]. From the negotiated throughput QoS value we can compute at what time interval `T-DATA requests` are expected by the service provider (cf. Figure 1). So, delaying the third `T-DATA request` in a sequence of `T-DATA requests` (`t1`, `t1+tr`, `t1+2tr+delay`) should cause aborting the transport connection. Since compulsory QoS semantics allows to do better than the negotiated value of the throughput QoS parameter, it should not cause problems sending to fast (at least from the testing point of view).

QoS assessment is different from OSI conformance testing since non-functional properties, like performance properties as discussed in the above example, have to be addressed too. However, in this paper we do not develop a QoS testing methodology in the sense that we give guidance for the specification of QoS test cases. Our emphasis is on a support for the specification of test cases. Parts of such a support are a QoS testing architecture and a test specification language. These aspects are discussed in the following sections.

4 QoS Testing Architecture

QoS testing extends protocol conformance testing approaches [12, 10] in several directions.

1. We have to deal with distributed IUTs which are controlled by several test components running in parallel. Multiple closely related data streams (e.g. voice, video, audio, graphical animation) have to be controlled. In order to cope with the (possibly) diverse QoS requirements per data stream, several test components have to be active concurrently. Furthermore, since some QoS parameters require synchronization of different data streams it becomes necessary that test components can communicate in order to synchronize their data streams they are handling.

2. The control of certain performance properties during a test requires to make predictions of the timing behavior of the different test components, the underlying network, and the communication infrastructure supporting communication between test components.

The OSI conformance testing methodology does not support the test of distributed IUTs. It only supports the definition of multi party testing architectures [11, 12], i.e., test architectures with several test components running in parallel.

TTCN, the test specification language for OSI conformance testing [13, 14], does not deal with timing constraints of test components, underlying network, or communication infrastructure. To a limited extend TTCN timers can be used to specify some time constraints, but this does not suffice in QoS testing. For testing performance related QoS parameters it is required to determine the exact times of reception of data packets and to correlate these values with previously determined values.

Figure 2 presents our ideas of a test architecture for QoS testing. The test architecture consists of *test components*, an *IUT*, and a *network facility*. The test components access the IUT via *service access points*. IUT and network facility are connected via *network interfaces*. Test components and network facility communicate by using *communication links*.

The test components are the active components in the architecture. They control the test and drive the IUT according to the test specification. The network facility provides an underlying network service which is necessary for the IUT in order to provide its service to the test components. The network facility might be a real network or just a network simulator. In order to drive the IUT into situations where QoS parameters cannot be guaranteed anymore, i.e., where the IUT should present a special behavior according to the QoS semantics, the test components may influence the behavior of the network facility.

The design of our QoS testing architecture has been motivated by the following design considerations:

- QoS negotiation and QoS maintenance are activities that involve service users and service provider.

- QoS maintenance is performed by the service provider. It is the service provider whose behavior in performing QoS maintenance functions is assessed during QoS testing. Therefore, in our case the IUT comprises all those entities which are in OSI terminology referred to as service provider.

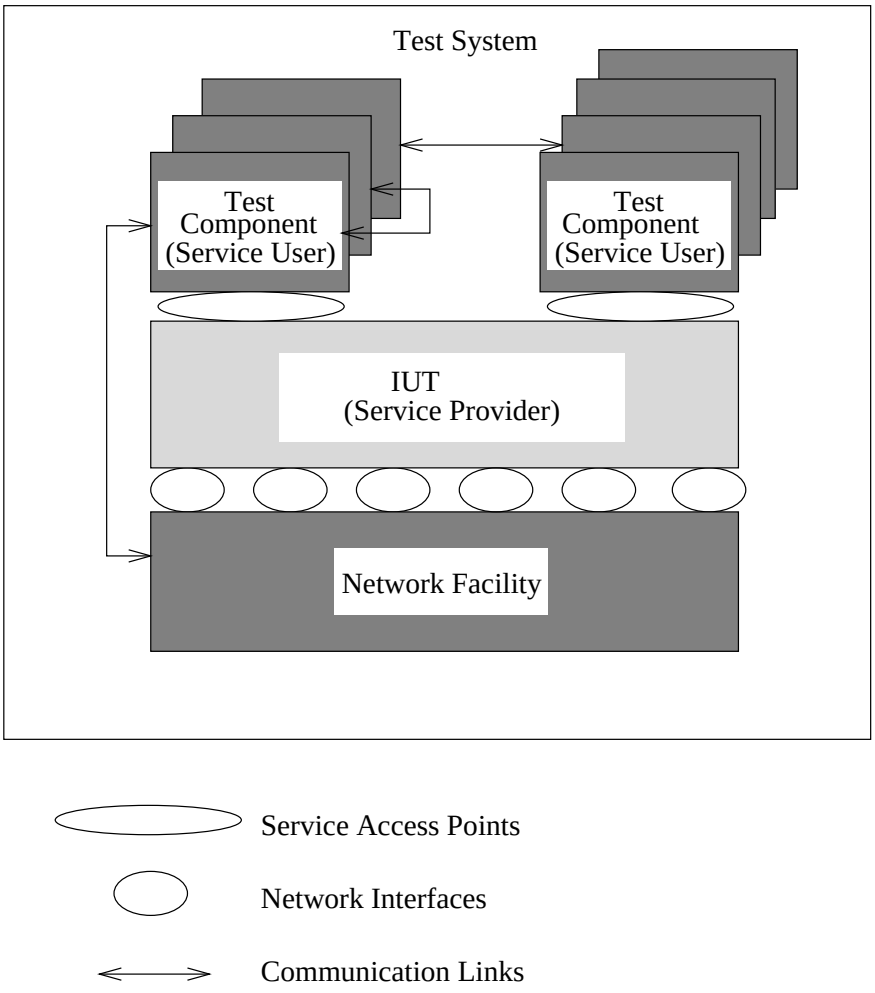

Figure 2: A testing architectue for QoS testing

- Similar to other testing architectures [12, 10] we assume that an IUT may be a small part of a bigger software system. But unlike to existing conformance testing architectures we presuppose that the IUT is driven only via service access points. In our model the parts in which the IUT might be embedded is hidden in the network facility. The network facility abstracts from the network that supports the communication between parts of the IUT. The network facility should be configurable in the sense that various parameters that determine the communication behavior of the network facility can be controlled. This enables us to change traffic characteristics like error rates or delays.

5 On the design of TelCom TSL

The definition of *TelCom TSL* has been influenced by our work on the formal definition of TTCN, concurrent TTCN [19, 20], QoS specification and verification [18, 17], and the specification and generation of TTCN test cases based on SDL and MSC [6, 7, 8, 9]. TelCom TSL should meet two requirements.

First, a test case specifier should define the functional behavior of a test case indepen-

dently of any timing constraints that may apply. Referring back to the discussion of the throughput QoS monitoring example (Section 3) the functional behavior of a possible test case would define the following behavior of test components, network facility and IUT: one test component (Figure 2) generates a sequence of `T-DATA request` service primitives. The `T-DATA request` service primitives are accepted and processed by the IUT which in turn hands over the processed `T-DATA request`s to the network facility. The network facility may transmit the `T-DATA request`s to a second entity of the IUT which eventually issues a `T-DATA indication` to a second test component.

Second, the test case specifier should be enabled to add timing requirements at any time in the test case design and validation process. For instance, the timing of `T-DATA request`s is dependent on the QoS throughput value which was negotiated among service users and service provider during connection establishment. The higher the throughput the shorter are the time intervals between successive `T-DATA request`s. The throughput does not affect the functional description of the test case, but may impose additional constraints on the timing behavior of network facility and test components.

In order to cope with the first requirement TelCom TSL provides the means for the description of the functional behavior of test systems (Figure 2) except for the IUT and network facility. The IUT is assumed to be a black box whose external behavior is visible only. The network facility is also assumed to be given. But unlike the IUT, the network facility is under the control of a test operator. Particularly, the test operator has the possibility to change the configuration of the network facility with respect to QoS characteristics. TelCom TSL structures a test system into a hierarchically ordered set of entities. We distinguish *test components*, *link processes* and the *test system* itself.

- Test components are active entities running in parallel. Test components have assigned a behavior specification that describe their behavior during execution of a test case. For the above discussed example, the behavior specification for the calling test component consists of a sequence of `T-DATA request`s with associated data parameter. The called test components behavior description simply consists of a sequence of corresponding `T-DATA indication`s. Furthermore, calling test component and called test component are prepared for receiving connection abort indications which are expected from the IUT when a violation of the QoS throughput value has been noticed.

- To make the distribution of test components over real systems explicit, we have introduced *test modules* which combine all test components located on one site. Test components located on the same system communicate synchronously.

- Communication between test components on different systems is supported by unidirectional link processes. Interaction between test components and link processes is synchronous. Link processes may also be used for the coordination among test components and between test components and network facility. For instance, a link process between calling and called test component can be used to inform the other side that a next service primitive has been initiated. Link processes are an abstraction of the testing communication infrastructure and may include hardware and software components.

IUT and test components communicate through *service access points*. IUT and network facility use *network interfaces* for communication purposes. The realization of these interfaces

is not constraint by TelCom TSL. The only requirement imposed is that communication among IUT, test components, and network facility is not arbitrarily delayed but that the delay is fixed and known. This constraint stems from the requirements that QoS testing imposes stringent timing constraints and therefore a certain knowledge of the timing behavior of system components is needed.

The behavior of entities is defined in terms of labeled transition systems [16]. Since execution of e.g., a `T-DATA request` on a real system requires a finite amount of time, we dropped the assumption that actions are instantaneous "without consuming time" [1]. Similarly, we assume that the delay introduced by link processes is finite and known, i.e., the time to transmit an amount of information between test components and the processing overhead is predictable.

In order to determine the timing behavior of a test system, the internal organization of the real system executing test components and link processes may also have to be considered. If a multiprocessor system supports the assignment of test processes and (if necessary) link processes to processors, we can assume that these processes are executed in parallel. The execution of processes sharing the same processor is modeled by an arbitrary interleaving of actions of the processes involved. Based on the knowledge of the timing behavior of all components of a test real system we are able to make predictions whether a given test case with given timing constraints can be executed correctly on a specific system, i.e., whether the intended result can be achieved.

As shown in Figure 1, a sequence of `T-DATA request`s that complies with the negotiated QoS throughput value requires that the calling test component issues every `tr` time units a `T-DATA request`. Assuming that calling test component, IUT, and network facility are executed on one single processor system without additional load, a possible distribution of processing time for performing the `T-DATA request` is shown in Figure 3 a). The time for performing the exchange of information between calling test component and IUT is included in the processing time of the calling test component. Similarly, the time for the interaction between IUT and network facility is included in the network facility processing time. A slightly different system architecture or a slightly changed load of the system may show another test result. In Figure 3 b) it is assumed that the execution of the IUT and network facility is delayed so that the theoretically predicted time for the next `T-DATA request` is missed.

The internal organization of the test system architecture and the timing constraints of link processes, interfaces and test components are to be seen as external parameters that need not be known while specifying the functional behavior of a test case for QoS testing. If actual values for these parameters are known (later in the test case design process) then validation of the timing behavior of the test case against the QoS timing requirements becomes possible.

6 Conclusions

QoS testing extents protocol conformance testing in several directions. Thus, a new QoS testing methodology is needed. We started to tackle this problem by presenting ideas for a new QoS testing architecture and TelCom TSL, a new test specification and implementation language for advanced telecommunications applications. Our test architecture is able to deal

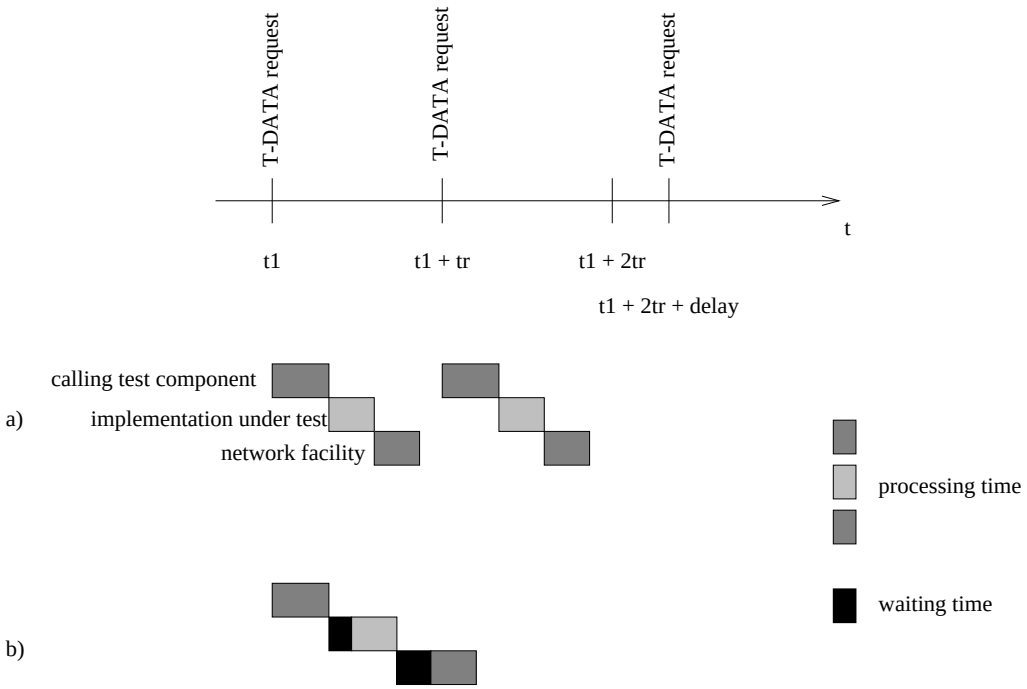

Figure 3: Throughput testing system elapsed time

with distributed IUTs and allows to control an underlying network facility with respect to QoS characteristics. The design ideas of TelCom TSL are influenced by our experience with TTCN and concurrent TTCN. During the next development steps we will refine our test architecture and define the concrete syntax and semantics of TelCom TSL. An application example will help to focus on the practical needs of QoS testing.

References

[1] T. Bolognesi, E. Brinksma. *Introduction to the ISO Specification Language LOTOS*, Computer Networks and ISDN Systems, Vol. 14, 1987.

[2] CCITT/ITU, *Data Communication Networks Open Systems Interconnection (OSI) Model and Notation, Service Definition*, X.200 - X.219, 1988.

[3] A. Danthine, Y. Baguette, G. Leduc, Léonard, *The OSI 95 Connection-Mode Transport Service - The Enhanced QoS*, A. Dantine, O. Spaniol (Editors), *High Performance Networking*, IFIP, 1992.

[4] A. Danthine, O. Bonaventure. *From Best Effort to Enhanced QoS*, CIO Deliverable R2060/ULg/CIO/DS/P/004/b1, 1993.

[5] ISO/ITU-T. *Formal Methods in Conformance Testing*, ITU-T TS SG10 Q8 and ISO SC21 WG1 P54 Southampton output, 1994.

[6] J. Grabowski, D. Hogrefe, R. Nahm. *Test Case Generation with Test Purpose Specification by MSCs.* In O. Faergemand and A. Sarma, editors, *SDL'93 - Using Objects.* North-Holland, October 1993.

[7] J. Grabowski, D. Hogrefe, R. Nahm, A. Spichiger. *Die* SAMSTAG *Methode und ihre Rolle im Konformitätstesten.* In *Praxis der Informationsverarbeitung und Kommunikation (PIK) Nr. 4/94*, K.G. Saur Verlag, München, December 1994.

[8] J. Grabowski, D. Hogrefe, I. Nussbaumer, A. Spichiger. *Improving the Quality of Test Suites for Conformance Tests by using Message Sequence Charts.* In *Proceedings of the 'Fourth European Conference on Software Quality' in Basel (Switzerland)*, October 1994.

[9] J. Grabowski, D. Hogrefe, I. Nussbaumer, A. Spichiger. *Combining MSCs and Data Descriptions in order to Generate Executable Test Cases for ISDN Systems.* In *Proceeding of the XV International Switching Symposium (ISS'95) - World Telecommunications Congress*, Berlin, April 95.

[10] D. Hogrefe *Conformance Testing Based on Formal Methods*, FORTE 90, North-Holland, 1990.

[11] ISO/IEC. *Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 1: General Concepts*, ISO/IEC 9646-1, 1994.

[12] ISO/IEC. *Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 2: Abstract Test Suite Specification*, ISO/IEC 9646-2, 1994.

[13] ISO/IEC. *Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN)*, ISO/IEC 9646-3, 1992.

[14] ISO/IEC. *Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN): Amendment 1: TTCN Extensions*, ISO/IEC 9646-3 DAM 1, 1993.

[15] H. Leopold, A. Cambell, D. Hutchinson, N. Singer. *Towards an Integrated Quality of Service Architecture (QoS-A) for Distributed Multimedia Communications*, A. Dantine, O. Spaniol (Editors), *High Performance Networking*, IFIP, 1992.

[16] G. Plotkin. *A Structural Approach to Operational Semantics*, Aarhus University, Computer Science Department, 1981.

[17] T. Plagemann, B. Plattner, M. Vogt, T. Walter. *A Model for Dynamic Configuration of Light-Weight Protocols*, Proceedings IEEE Third Workshop on Future Trends of Distributed Computing Systems, IEEE, 1992.

[18] J. Montiel, E. Rudolph, J. Burmeister (Editors), *Methods for QoS Verification and Protocol Conformance Testing in IBC - Application Guidelines* -, TOPIC, 1993.

[19] T. Walter, J. Ellsberger, F. Kristoffersen, P.v.D. Merkhof. *A Common Semantics Representation for SDL and TTCN*, Proceedings PSTV XII, North-Holland, 1992.

[20] T. Walter, B. Plattner. *An Operational Semantics for Concurrent TTCN*, Proceedings PTS V, North-Holland, 1992.