# An Overview of OSI Conformance Testing

Jan Tretmans
Formal Methods & Tools group
University of Twente

January 25, 2001

## 1 Introduction

The development of distributed systems, in which the computer functionality, such as processing functions, information storage, and human interaction, is distributed over different computer systems, raises the need for exchanging information between these systems. To have computer systems communicate successfully, the communication must occur according to well-defined rules. A *protocol* describes the rules with which computer systems have to comply in their communication with other computer systems. A *protocol entity* is that part of a computer system that takes care of the local responsibilities in communicating according to the protocol.

To have successful communication among computer systems, also from different manufacturers, many protocols are not developed in isolation, but within groups of manufacturers and users, with the aim of standardizing such protocols. This has led for instance to the development of the OSI Reference Model for Open Systems [ISO84], which serves as a framework for a set of standards that enable computer systems to communicate. However, to assure successful communication it is not sufficient to specify and standardize communication protocols. Implementations of these protocol standards are required for which it must be ascertained that these implementations really behave according to these standards protocol specifications, i.e., *conform* to these standards. One way to do this is by *testing* these protocol implementations. This activity is known as *protocol conformance testing*.

This note gives an introduction into some of the important concepts of protocol conformance testing. It is largely based on the standard ISO 9646: "Conformance Testing Methodology and Framework" [ISO91]. This standard was originally developed to give a framework and define common terminology for testing of OSI systems. Although OSI protocols themselves are not often used anymore the concepts for testing their implementations have a broader applicability and can, and actually are, also used in testing of other kinds of protocol systems.

This note starts with a discussion of what conformance testing is, after which the main phases and aspects of conformance testing according to [ISO91] are presented.

# 2 Conformance Testing

Testing is the process of trying to find errors in a system implementation by means of experimentation. The experimentation is usually carried out in a special environment, where normal and exceptional use is simulated. The aim of testing is to gain confidence that during normal use the system will work satisfactory: since testing of realistic systems can never be exhaustive, because systems can only be tested during a restricted period of time, testing cannot ensure complete correctness of an implementation. It can only show the presence of errors, not their absence.

Protocol conformance testing is a kind of testing where an implementation of a protocol entity is tested with respect to its specification. The aim is to gain confidence in the correct functioning of the implementation with respect to a given specification, and hence to improve the probability that the protocol implementation will communicate successfully with peer protocol entities.

To conduct testing, experiments, or tests must be systematically devised. These tests are applied to an implementation, and the test outcomes are compared with the expected or calculated outcomes. Based on the results of the comparison a verdict can be formulated about the correctness of the implementation, which, if negative, can be used for improving the implementation.

In testing, in particular in software testing (see e.g., [Mye79, Whi87]), a distinction is made between functional testing and structural testing. *Structural testing*, also referred to as *white-box testing*, is based on the internal structure of a computer program. The aim is to exercise thoroughly the program code, e.g., by executing each statement at least once, or by trying to execute all paths through the program code taking into account decisions, branches, loops, etc. Tests are derived from the program code. Structural testing is most used in the early stages of program development. With *functional testing* the emphasis is on testing the externally observed functionality of a program based on its specification. Functional testing is also called *black-box testing*: a system is treated as a black box, whose functionality is checked by observing it, i.e., no reference is made to the internal structure of the program. The main goal is to determine whether the right (with respect to the specification) product has been built. Functional tests are derived from the specification. Consequently, the most important prerequisite is a precise, complete and clear specification. Functional testing is usually concentrated in the later stages of program development.

Protocol conformance testing is a kind of functional testing: an implementation of a protocol entity is solely tested for conformance with respect to the requirements given in its specification. The idea is that only systems with correctly implemented protocols can communicate successfully with peer entities. Often the specification is (internationally) standardized and then the goal is to *certify* the implementation with respect to the standard. Only the observable behaviour of the protocol implementation is tested, i.e., the interactions of an implementation with its environment; no reference is made to the internal structure of the protocol implementation. e.g., memory consumption. In practical conformance testing the internal structure of the entity is usually not even accessible to the tester: the computer system in which the entity under test is located need not be accessible, e.g., when testing is performed by an independent, accredited test laboratory, that has no access to the implementation details of an implementation.

**Conformance testing in the development trajectory** Conformance testing is only concerned with checking a protocol implementation, i.e., a software product (executing code), with respect to its specification. This implies that a specification must be available and, moreover, that the specification is correct and valid. Checking the correctness of a specification is referred to as *protocol validation*. It involves checking that the specification indeed prescribes the intended behaviour. For testing, validity of the specification is assumed; it is not the topic of conformance testing: if the specification contains a design error then, if the conformance testing process is correctly performed, each conforming implementation will have the same error.

**Other kinds of protocol testing** Since in practice it turns out that functional testing of an implementation in isolation, i.e., conformance testing, does not guarantee successful communication between systems, products are also tested in a realistic environment, for example in a model of a communication network. In this kind of testing the interaction with other computer systems can be examined in more detail. It is referred to as *interoperability testing*.

Apart from testing the functional behaviour of a protocol implementation, other kinds of testing test other aspects of a protocol, e.g., *performance testing* to measure the performance characteristics of an implementation, *robustness testing* to examine the implementation's behaviour in an erroneously behaving environment, and *reliability testing* to check whether the implementation continues to work correctly during a certain period of time.

**Parties involved** Conformance testing can be performed by different parties. First, the implementer or supplier of a product tests its product before selling it. Users of products, or their representative organizations, test products for their correct functioning. Telecommunications administrations check products before connecting them to their networks to prevent malfunctioning of a network caused by incorrectly implemented products. Finally, independent third party test laboratories can perform conformance tests for any of the previously mentioned parties. A system of accreditation allows testing laboratories to certify implementations that they have tested and judged to be conforming. Certification by accredited testing laboratories makes repeated testing by supplier, buyer, and network owner superfluous. Also repetition of tests by different network operators in different countries is not necessary if we can rely on testing having been performed according to well-defined procedures and standards.

**Standardization of conformance testing** If implementations of the same (internationally) standardized protocol are tested it should not occur that different test laboratories decide differently about conformance of the same implementation. Ideally, it should not be necessary that the same product is tested more than once by different testing laboratories. This is possible if testing is based on generally accepted principles, using generally accepted tests, and leading to generally accepted test results. To achieve this the International Organization for Standardization (ISO), together with the CCITT (now ITU-T), has developed a standard for conformance testing of Open Systems. This is the standard ISO IS-9646: "OSI Conformance Testing Methodology and Framework" [ISO91].

The purpose of this standard is 'to define the methodology, to provide a framework for specifying conformance test suites, and to define the procedures to be followed during

testing', leading to 'comparability and wide acceptance of test results produced by different test laboratories, and thereby minimizing the need for repeated conformance testing of the same system' [ISO91, part 1, Introduction]. The standard does not specify tests for specific protocols, but it defines a framework in which such tests should be developed, and it gives directions for the execution of such tests. The standard recommends that sets of tests, called *test suites*, be developed and standardized for all standardized protocols.

# 3 Overview of IS-9646

The current practice of protocol conformance testing is based on the standard ISO IS-9646, "OSI Conformance Testing Methodology and Framework" [ISO91, Ray87]. This standard defines a methodology and framework for protocol conformance testing assuming that protocols are specified using a natural language. It was originally developed for OSI protocols, but it is also used for testing other kinds protocols, e.g., ISDN and ATM protocols. The standard consists of five parts, each defining an aspect of conformance testing:

- part 1 is an introduction and deals with the general concepts;
- part 2 describes the process of abstract test suite specification;
- part 3 defines the test notation TTCN;
- part 4 deals with the execution of tests;
- part 5 describes the requirements on test laboratories and their clients during the conformance assessment process.

An overview of IS-9646 is given with most attention devoted to parts 1 and 2, i.e., to the generation and specification of test suites.

## 3.1 The Conformance Testing Process

In the process of conformance testing three phases are distinguished [ISO91, Part 1, Section 1.3]. They are depicted in Figure 1, together with the activity of protocol implementation. The first phase is the specification of an *abstract test suite* for a particular (OSI) protocol. We refer to it by *test generation* or *test derivation*. This test suite is abstract in the sense that tests are developed independently of any implementation. It is intended that abstract test suites of standardized protocols are standardized themselves. The second phase consists of the realization of the means of executing specific test suites. It is referred to as *test implementation*. The abstract test cases of the abstract test suite are transformed into executable tests that can be executed or interpreted on a real testing device or test system. The peculiarities of the testing environment and the implementation, which during testing is called *IUT* (Implementation Under Test), are taken into account. The last phase is the *test execution* The implemented test cases are executed with a particular IUT and the resulting behaviour of the IUT is observed. This leads to the assignment of a verdict about conformance of the IUT with respect to the standard protocol specification. The results of the test execution are documented in the *protocol conformance test report* (PCTR).

In the next subsections these phases are described in more detail. This leads to a more

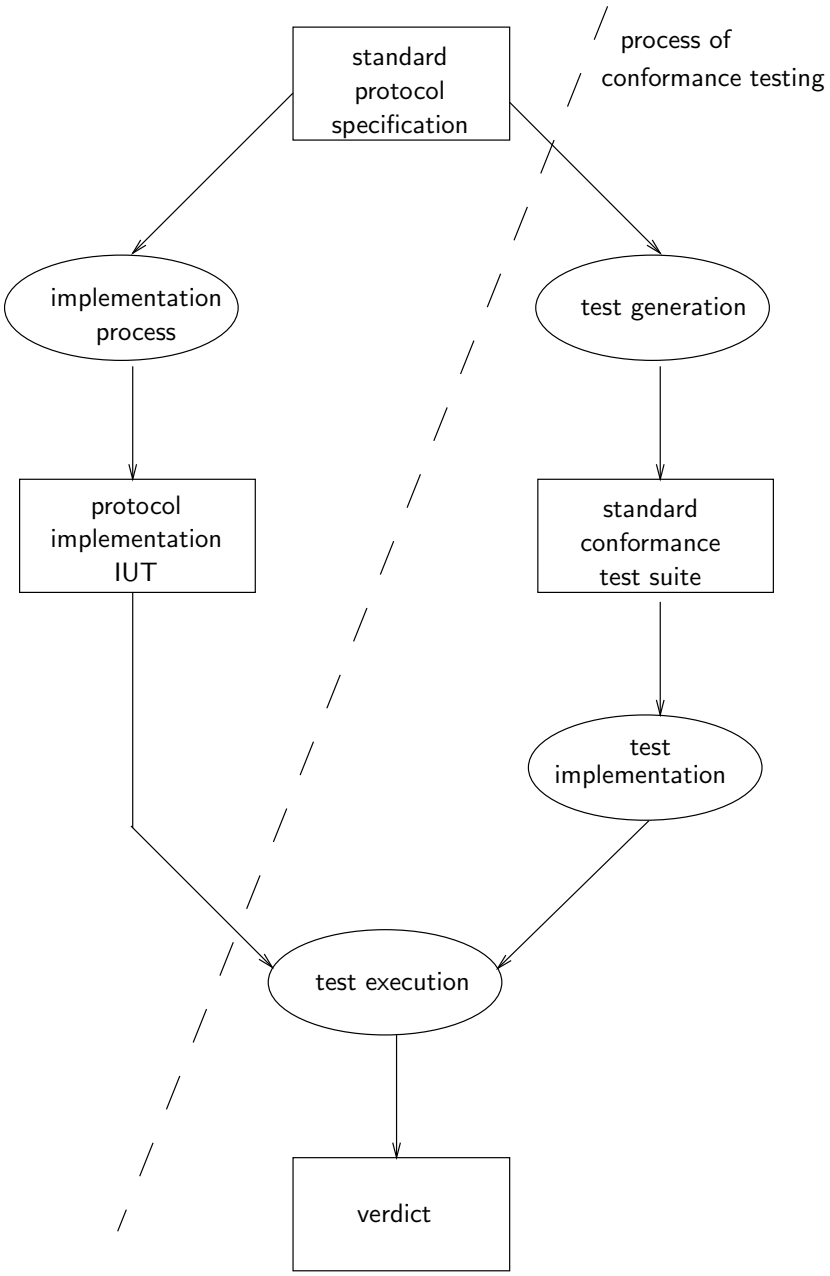


Figure 1: Global overview of the conformance testing process.

detailed view of the conformance testing process given in Figure 2.

## 3.2 A Conforming Implementation

Before an implementation can be tested for conformance it must be defined what conformance is: What does it mean that an implementation conforms to its specification? The definition of what constitutes a conforming implementation determines what should be tested. IS-9646 states that a system 'exhibits conformance if it complies with the conformance requirements of the applicable ... standard' [ISO91, Part 1, Section 5.1]. This means that a correct implementation is one which satisfies all conformance requirements, and that these conformance requirements must be mentioned explicitly in the protocol standard. Conformance requirements express what a conforming implementation shall do (positively specified requirements), and what it shall not do (negatively specified requirements).

A complication arises by the fact that a protocol standard does not uniquely specify one protocol, but a class of protocols. Most standards leave open a lot of options, which may or may not be implemented in a particular protocol implementation, but which, if implemented, must be implemented correctly. An implementer selects a set of options for implementation. All implemented options of a specific protocol implementation are listed by the implementer in the *PICS*, the *Protocol Implementation Conformance Statement*, so that the tester knows which options have to be tested. To assist in producing the PICS a *PICS proforma* is attached to the protocol standard. This is a questionnaire in which all possibilities for the selection of options are enumerated.

Restrictions on the selection of options are given in the *static conformance requirements* of a standard. They define requirements on the minimum capabilities that an implementation shall provide, and on the combination and consistency of different options.

**Example 3.1**
In the ISO/OSI Transport Protocol [ISO86] five classes (0 .. 4) are distinguished. In a particular implementation not all classes need be implemented. However, the choice is not completely free, e.g., if class 4 is implemented also class 2 must be implemented. Such a restriction is recorded in the protocol standard as part of the static conformance requirements. In the PICS the implemented classes of a particular implementation are documented.

□

The main part of a protocol standard consists of *dynamic conformance requirements*. They define requirements on the observable behaviour of implementations in the communication with their environment. They concern the allowed orderings of observable events, such as sending and receiving of PDUs (protocol data units) and ASPs (abstract service primitives), the coding of information in the PDUs, and the relation between information content of different PDUs.

**Example 3.2**
A dynamic conformance requirement of the ISO/OSI Transport Protocol is the requirement that after receiving a *T-PDU-connect-request* from the peer entity either the user of the Transport entity is notified by means of a *T-SP-connect-indication* service-primitive, or a *T-PDU-disconnect-request* is sent to the peer entity.

□

Summarizing, the definition of a conforming implementation is [ISO91, Part 1, Sections

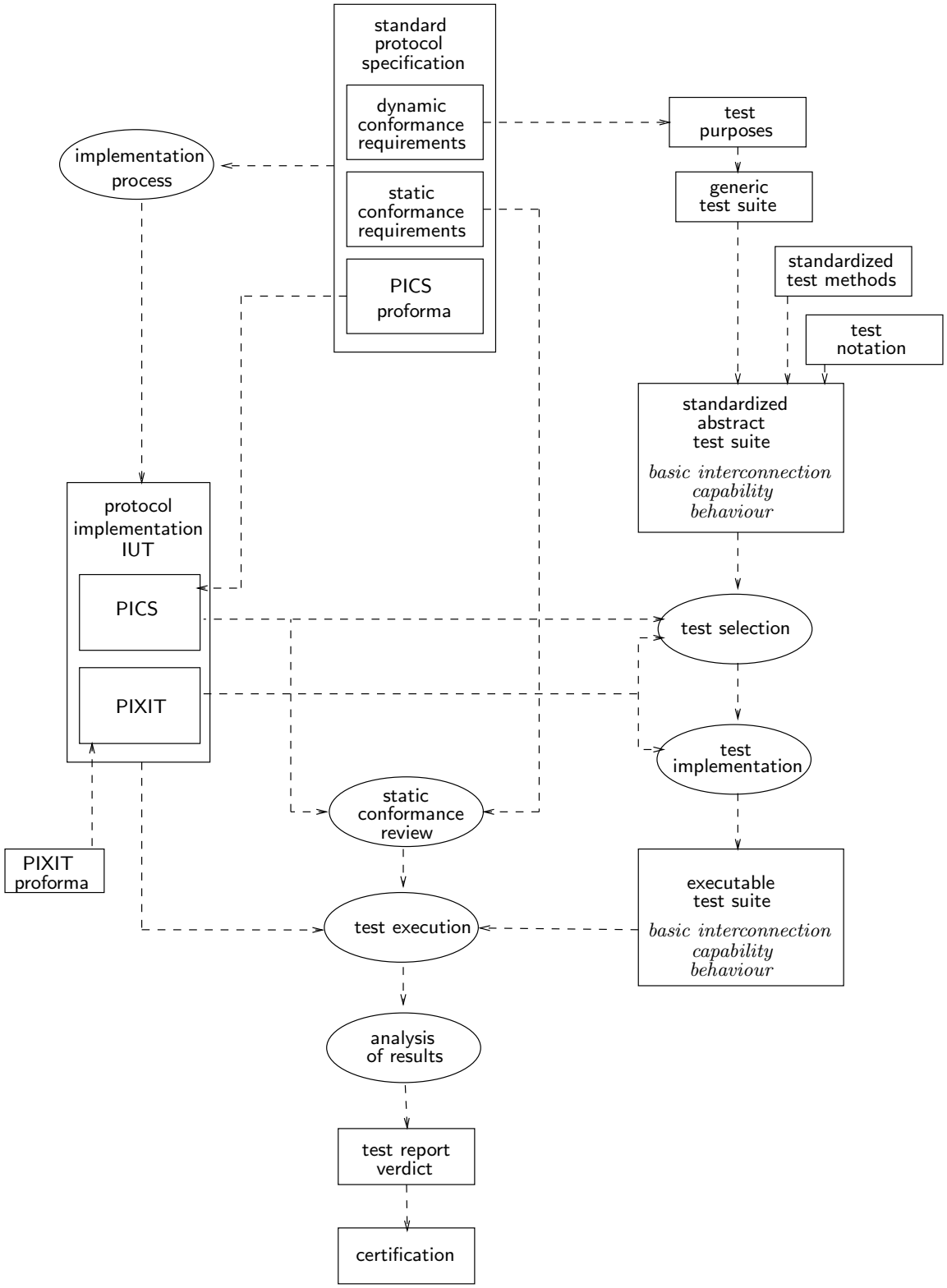


Figure 2: Detailed overview of the conformance testing process.

3.4.10 and 5.6]:

> 'A conforming implementation is one which satisfies both static and dynamic conformance requirements, consistent with the capabilities stated in the PICS.'

Conformance testing consists of checking whether an IUT satisfies all static and dynamic conformance requirements. For the static conformance requirements this means a reviewing process of the PICS delivered with the IUT. This is referred to as the *static conformance review*. For the dynamic conformance requirements this means running a number of tests against the IUT. The specification of one test is referred to as a *test case*. A *test suite* is a complete set of test cases, i.e., a set that tests all dynamic conformance requirements.

## 3.3 Test Generation

The first phase of the conformance testing process is *test generation*. It consists of systematically deriving test cases from a protocol specification. The goal is to develop an abstract test suite, i.e., a specification of a test suite that is implementation independent, specified in a well-defined test notation language, suitable for standardization, and testing all aspects of the protocol in sufficient detail. Since the relevance of a protocol specification with respect to conformance testing is its set of conformance requirements, and since the static conformance requirements are checked by reviewing the PICS, this means that the set of the dynamic conformance requirements in a protocol standard is the starting point for test generation.

Test cases are derived systematically from the dynamic conformance requirements in a multi-step procedure. In the first step, one or more *test purposes* are derived for each conformance requirement. A test purpose is a precise description of what is going to be tested in order to decide about the satisfaction of a particular conformance requirement. As the next step it is recommended to derive a *generic test case* for each test purpose. A generic test case is an operationalization of a test purpose, in which the actions necessary to achieve the test purpose are described on a high level, without considering a test method or the environment in which the actual testing will be done. The last step is the derivation of an *abstract test case* for each generic test case. In this step a choice is made for a particular *test method*, and the restrictions implied by the environment in which testing will be carried out are taken into account.

### Test methods

A protocol standard specifies the behaviour of a protocol entity at the upper and lower access points of the protocol ((N)-SAP en (N-1)-SAP). Hence the ideal points to test the entity are these SAPs. However, these SAPs are not always directly accessible to the tester. The points where the tester controls and observes the IUT are called the *Points of Control and Observation* (PCO). PCOs may, but need not coincide with the boundaries of the IUT. Normally in protocol conformance testing there are two PCOs, one corresponding with the upper access point of the IUT, and one with the lower access point. A similar conceptual separation is made for the tester. The part of the tester that controls and observes the PCO connected to the upper access point is called the *Upper*

*Tester* (UT). The part that controls and observes the PCO connected to the lower access point is called the *Lower Tester* (LT).

A *test method* defines a model for the accessibility of the IUT to the tester in terms of PCOs and their place within the OSI reference model [ISO84]. Aspects that can be distinguished are:

- existence of PCOs: if one of the access points is not accessible at all there is no PCO for that access point;
- whether there are other protocol layers between the PCO and the access point, and the kind of events that are communicated (ASPs or PDUs);
- the positioning of the PCOs in the same computer system as the IUT, called the *System Under Test* (SUT);
- the internal functioning of the tester in terms of the distribution of testing functions over LT and UT, and the rules that define their coordination: the *test coordination procedures*.

By varying these aspects different test methods are obtained. Some have been identified and standardized in IS-9646 [ISO91, Part 2, Section 12] for use in standardized abstract test suites. The basic configuration is Local Single-layer test method (LS-method), see Figure 3. In all standardized test methods the lower access point of the IUT is always accessible, usually via an underlying service; the upper access point may be hidden. Standardized test methods, apart from the LS-method, are the Distributed Single-layer test method (DS-method), the Coordinated Single-layer test method (CS-method), and the Remote Single-layer test method (RS-method). As another example, Figure 4 shows the DS-method. There are two PCOs. An example of a test method with one PCO is the RS-method: in the RS-method there is no upper tester.

A standardized abstract test suite refers to a particular test method, choosing the most appropriate one.

The four test methods that were mentioned can be used in variations where the IUT consists of more than one subsequent protocol layers. These layers can be tested as a whole (multi-layer testing), or one layer can be tested embedded in the other layers (embedded testing). The test methods are LM, CM, DM and RM (Local Multi-layer, etc.), and LSE, CSE, DSE and RSE (Local Single-layer Embedded, etc).

**Test notation**

Since abstract test suites are standardized, they must be specified in a *test notation* that is well-defined, independent of any implementation, and generally accepted. IS-9646 recommends the semi-formal language *TTCN*, the *Tree and Tabular Combined Notation*, which is defined in [ISO91, part 3] and more recently in [ISO97]. (A major revision leading to TTCN Version 3 is expected to appear soon [GH99]).

In TTCN the behaviour of test cases is specified by sequences of input and output events that occur at the PCOs. A sequence can have different alternatives, where different subsequent behaviours can be chosen, e.g., depending on output produced by the IUT, the expiration of timers, or values of internal parameters of the tester. Successive events are indicated by increasing the level of indentation, alternative events have the same indentation. A sequence ends with the specification of the verdict that is assigned when the execution of the sequence terminates. The verdicts in the different possible alternative

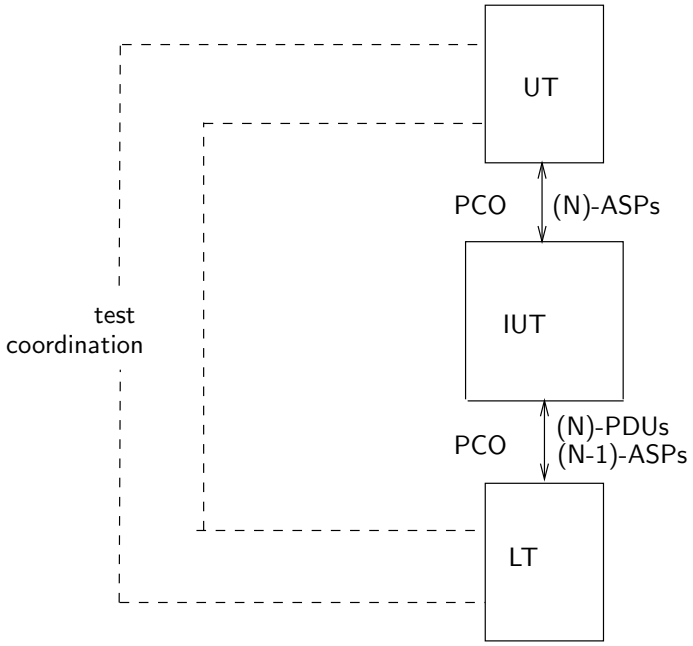


Figure 3: The local test method.

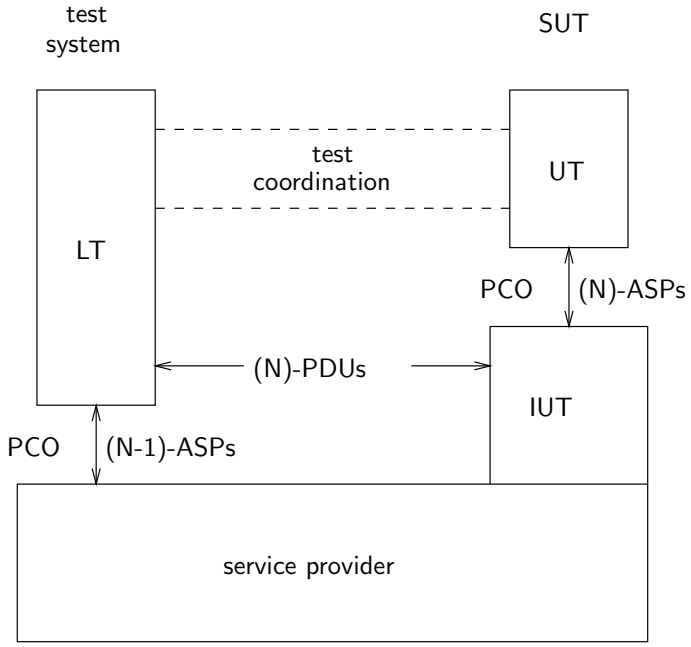


Figure 4: The distributed test method.

behaviours differ. Some alternatives will describe correct behaviour, ending with the positive verdict **pass**, while other alternatives describe erroneous behaviour, ending with the negative verdict **fail**. The verdict **inconclusive** indicates correct but not intended behaviour, see Section 3.5.

TTCN is defined in such a way that automatic execution is feasible. A simplified example of a TTCN behaviour is presented in Figure 5. More about TTCN, apart from the defining standard, can be found in [PM92].

| **Test Case Dynamic Behaviour** | | |
|---|---|---|
| **Test Case Name:** | Conn_Estab | |
| **Group:** | transport/connection | |
| **Purpose:** | Check connection establishment with remote initiative | |
| behaviour | constraints | verdict |
| + preamble | | |
| LT ! T-PDU-connect-request | | |
| UT ? T-SP-connect-indication | | |
| UT ! T-SP-connect-response | | |
| LT ? T-PDU-connect-confirm | | **pass** |
| OTHERWISE | | **fail** |
| LT ? T-PDU-disconnect-request | | **inconclusive** |
| OTHERWISE | | **fail** |

Figure 5: A simplified TTCN example.

### Classification of tests

Tests can be classified according to the extent to which they give an indication of conformance. The following distinction is made:

- basic interconnection tests
- capability tests
- behaviour tests
- conformance resolution tests

The classification is applicable to generic, abstract and the executable tests, which will be discussed in Section 3.4.

Basic interconnection tests are used to guarantee a basic level of interconnection between the tester and the IUT. Their main purpose is economical: before an expensive test environment is developed first some basic functions of the IUT are checked, e.g., the establishment of a connection between the tester and the IUT.

Capability tests serve to verify the compliance between the implemented options and the options stated in the PICS.

Behaviour tests constitute the main part of a test suite. They test the dynamic conformance requirements of a protocol standard in full detail within the limits of technical and economical feasibility. They are the basis for the final verdict about conformance.

Conformance resolution tests do not belong to the actual conformance tests. They form supplementary tests that can be used to do extra testing if problems are encountered, or to trace errors. These tests have a heuristic nature, they are not standardized, and they

cannot be used as a basis for the final verdict.

#### Hierarchical structuring of tests

A test suite is a complete set of tests for conformance testing of a particular protocol. Elements of a test suite are tests, or test cases. A test case specifies one experiment, related to one test purpose and to one conformance requirement. Related test cases can be grouped into test groups with corresponding test group objective. Grouping can occur at different levels.

Within a test case test steps and test events can be distinguished. A test event is one interaction at a PCO, e.g., sending or receiving one PDU. A test step groups successive test events. An example of a test step is a preamble: a sequence of events that brings the IUT in a state from which the body of the test case that tests the test purpose can be tested. Analogously the postamble test step brings the IUT back to a specified state, e.g., the initial state, after the main part of a test case has been executed.

The hierarchical structuring is applicable to all levels of test cases. Also conformance requirements and test purposes can be grouped.

### 3.4 Test Implementation

Starting point for test implementation is the (standardized) abstract test suite. The abstract test suite is specified independently of any real testing device. In the test implementation phase it is transformed into an *executable test suite*, i.e., a test suite which can be run on a specific testing device with a specific IUT.

Before starting to implement, a selection from the abstract test suite must be made. The abstract test suite contains all possible tests for a particular protocol, for all possible options. It does not make sense to test for options that are not implemented according to the PICS. Therefore the tests relevant to the IUT are selected based on the PICS. In IS-9646 this is called *test selection*[1].

The PICS contains protocol dependent information. To derive executable tests this is insufficient; also information about the IUT and its environment must be supplied. Such information is called PIXIT (Protocol Implementation eXtra Information for Testing). The PIXIT may contain address information of the IUT, or parameter and timer values which are necessary to implement the test suite. The PIXIT, like the PICS, is supplied by the supplier of the IUT to the testing laboratory. To guide production of the PIXIT the testing laboratory provides a *PIXIT proforma*.

The selected and implemented test cases with parameter values according to the PIXIT form the executable test suite, which can be executed on a real tester or test system. During implementation care must be taken that the tests are implemented correctly, according to the semantics of the test notation used for the specification of the abstract test suite.

[1] Note that the notion of 'test selection' is sometimes used in a different way, viz. as selecting from an (infinite) set of possible (automatically generated) test cases.

### 3.5 Test Execution

During the test execution phase a specific IUT is actually tested leading to a verdict about conformance of the IUT. The first step consists of the static conformance review: the PICS of the IUT is checked for internal consistency and compared with the static conformance requirements of the standard. The second step consists of executing the executable test cases on a real tester. The reactions of the IUT are observed and compared with the reactions specified in the test case. For each test case a verdict is assigned. The verdict is either *pass*, *fail*, or *inconclusive*. *Pass* indicates that the test was executed successfully, and that the goal expressed in the corresponding test purpose was achieved. *Fail* indicates that the implementation does not conform to the specification. *Inconclusive* indicates that no evidence of non-conformance was found, but that the test purpose was not achieved.

**Example 3.3**
Suppose the test purpose in the TTCN example in Figure 5 is to check the correct connection establishment of the Transport Protocol, by testing the sequence of actions *T-PDU-connect-request*, *T-SP-connect-indication*, *T-SP-connect-response*, *T-PDU-connect-confirm*. If the IUT reacts with *T-PDU-disconnect-request* after having received *T-PDU-connect-request*, the verdict *inconclusive* is assigned: this behaviour is allowed according to the Transport standard, but the the verdict *pass* cannot be assigned since the test purpose was not achieved.

□

Finally, the results of the static conformance review and the verdicts of all test cases are combined, leading to a verdict about conformance of the IUT with respect to the protocol specification. Normally the final verdict is *pass* if and only if no individual test resulted in the verdict *fail*. All results, including the final verdict, are documented in the PCTR (Protocol Conformance Test Report).


## Acknowledgements

Jeroen van de Lagemaat (University of Twente) is acknowledged for contributing to the original, Dutch version of this paper [TL91].


# References


[GH99] J. Grabowski and D. Hogrefe. Towards the Third Edition of TTCN. In G. Csopaki, S. Dibuz, and K. Tarnay, editors, *Int. Workshop on Testing of Communicating Systems 12*, pages 19–29. Kluwer Academic Publishers, 1999.

[ISO84] ISO. *Information Processing Systems, Open Systems Interconnection, Basic Reference Model*. International Standard IS-7498. ISO, Geneve, 1984.

[ISO86] ISO. *Information Processing Systems, Open Systems Interconnection, Connection Oriented Transport Protocol Specification*. International Standard IS-8073. ISO, 1986.

[ISO91] ISO. *Information Technology, Open Systems Interconnection, Conformance Testing Methodology and Framework*. International Standard IS-9646. ISO, Geneve, 1991. Also: CCITT X.290–X.294.

[ISO97] ISO/IEC. *Information Technology – Open Systems Interconnection – Conformance Testing Methodology and Framework – Part 3: The Tree and Tabular Combined Notation (TTCN).* International Standard ISO/IEC 9646-3. ISO/IEC, Geneve, 1997. Second Edition.

[Mye79] G.J. Myers. *The Art of Software Testing.* John Wiley & Sons Inc., 1979.

[PM92] R.L. Probert and O. Monkewich. TTCN: The international notation for specifying tests for communications systems. *Computer Networks and ISDN Systems*, 23(5):417–438, 1992.

[Ray87] D. Rayner. OSI conformance testing. *Computer Networks and ISDN Systems*, 14:79–98, 1987.

[TL91] J. Tretmans and J. van de Lagemaat. *Handboek Telematica*, volume II, chapter Conformance Testen, pages 4400 1–19. Samson, 1991. In Dutch. Also: Memorandum INF-90-86, University of Twente, The Netherlands.

[Whi87] L. J. White. *Software Testing and Verification*, volume 26 of *Advances in Computers.* Academic Press, 1987.