ETSI TECHNICAL REPORT

ETR 190

August 1995

Source: ETSI TC-MTS

Reference: DTR/MTS-00021

ICS: 33.020

Key words: Test specification, standardisation methodology, conformance testing, modularization, re-use of test suites

Methods for Testing and Specification (MTS); Partial and multi-part Abstract Test Suites (ATS); Rules for the context-dependent re-use of ATSs

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

Contents

Foreword

This ETSI Technical Report (ETR) was produced by the Methods for Testing and Specification (MTS) Technical Committee of the European Telecommunications Standards Institute (ETSI).

ETRs are informative documents resulting from ETSI studies which are not appropriate for European Telecommunication Standard (ETS) or Interim European Telecommunication Standard (I-ETS) status. An ETR may be used to publish material which is either of an informative nature, relating to the use or the application of ETSs or I-ETSs, or which is immature and not yet suitable for formal adoption as an ETS or an I-ETS.

Blank page

1 Scope

The objective of this ETSI Technical Report (ETR) is to give guidance on the creation of re-usable Abstract Test Suites (ATSs) by providing some rules and associated examples.

The main subject is the abstract specification level, but to some extent tool functionality's are discussed and maintenance and libraries of test specifications are considered, too.

The guidance given by the rules and the examples is aimed at standardisation bodies developing conformance testing standards, and also to organisations, like Conformance Testing Services (CTS) programmes, pre-standardisation institutions and private institutions, which develop conformance testing standards with the objective to contribute them to European Standardisation.

Some rules are applicable to situations where new ATSs are created. Some other rules are dedicated to the modularization and the re-use of existing ATSs, particularly in the profile testing context.

Before actually stating the rules, the general needs of test specifiers are discussed in clause 4, some known solution attempts of the past are presented in clause 5 and the problem is put in a general scheme in clause 6. The actual rules and examples are collected in clause 7. Finally, clause 8 summarises the possible effects of the discussions in the previous clauses on the test specification methodology.

Parallel to the development of this ETR, an extension of Tree and Tabular Combined Notation (TTCN) to support the modularization of Abstract Test Specifications has been discussed among experts. This extension of TTCN contains the concept of TTCN Modules, and import/export features for those Modules. There is still no unique opinion on what the new concept will exactly look like. annex C of this ETR gives a list of open issues in that concept. The rules and examples on re-use of testing material given in this ETR are partially based on the new modularization concept, and are also partially independent of this. Where assumptions have been made on open issues, this is indicated in the text.

Except for the reference to Modular TTCN, this ETR does not contradict the references listed in clause 2.

Three annexes conclude the ETR. For the purpose of quick cross-reference, they provide the list of rules, the list of examples and the list of most important open issues in Modular TTCN, respectively.

2 References

This ETR incorporates by dated or undated references, provisions from other publications. These references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this ETR only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

[1] ISO/IEC 9646-1 (1994): "Information technology - OSI conformance testing methodology and framework - Part 1: General concepts".

[2] ISO/IEC 9646-2 (1994): "Information technology - OSI conformance testing methodology and framework - Part 2: Abstract test suite specification".

[3] ISO/IEC 9646-3 (1992): "Information technology - OSI conformance testing methodology and framework - Part 3: Tree and tabular combined notation".

[4] ISO/IEC 9646-3 AM. 1 (1993): "Information technology - OSI conformance testing methodology and framework - Part 3: Tree and tabular combined notation".

[5] ISO/IEC 9646-3/PDAM 2 (December 1993): "Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: TTCN - Amendment 2 - Further Extensions".

[6] ISO/IEC 9646-4 (1994): "Information technology - OSI conformance testing methodology and framework - Part 4: Test realisation".

[7] ISO/IEC 9646-5 (1994): "Information technology - OSI conformance testing methodology and framework - Part 5: Requirements on test laboratories and clients for the conformance assessment process".

[8] ISO/IEC 9646-6 (1994): "Information technology - OSI conformance testing methodology and framework - Part 6: Protocol Profile Test Specification".

[9] ISO/IEC 9646-7 (1995): "Information technology - OSI conformance testing methodology and framework - Part 7: Implementation Conformance Statements".

[10] ISO/IEC 8882-3 (1993): "Information technology - Telecommunications and information exchange between systems - X.25 DTE conformance testing, Part 2: Packet layer conformance test suite".

[11] ETR 141 (1994): "Methods for testing and specification (MTS) Protocol and profile conformance testing Specifications: The Tree and Tabular Combined Notation (TTCN) style guide".

[12] ETS 300 655: "Signalling Protocols and Switching (SPS); ASN.1 library definition".

NOTE 1: This ETS is presently in a draft (STC) format and is expected to be published in mid-1996.

[13] ETR 210: "Signalling Protocols and Switching (SPS); ASN.1 library index".

NOTE 2: This ETR is presently in a draft (STC) format and is expected to be published in mid-1996.

[14] ETS 300 406 (1995): "Methods for testing and specification (MTS) Protocol and profile conformance testing specifications: Standardisation methodology".

[15] ETS 300 080-1 (1994): "Integrated Services Digital Network (ISDN); ISDN lower layer protocols for telematic terminals; Part 1: Profile and Requirements List (RL)".

NOTE 3: This ETS is presently in a draft (STC) format and is expected to be published in mid-1996.

[16] ETS 300 080-2 (1994): "Integrated Services Digital Network (ISDN); ISDN lower layer protocols for telematic terminals; Part 2: Profile Test Specification Summary (PTS-S)".

NOTE 4: This ETS is presently in a draft (STC) format and is expected to be published in mid-1996.

[17] ETS 300 080-3 (1994): "Integrated Services Digital Network (ISDN); ISDN lower layer protocols for telematic terminals; Part 3: Profile Specific Test Specification (PSTS)".

NOTE 5: This ETS is presently in a draft (STC) format and is expected to be published in mid-1996.

[18] ETS 300 080-4 (1994): "Integrated Services Digital Network (ISDN) - ISDN lower layer protocols for telematic terminals, Part 4: Profile System Conformance Test Report (SCTR)".

NOTE 6: This ETS is presently in a draft (STC) format and is expected to be published in mid-1996.

[19] EWOS/ETG 022 (1992): "Test specifications for embedded protocols in application profiles".

[20] ISO/IEC 8073 (1992): "Information technology - OSI Protocol for providing the connection-mode transport service".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of this ETR, the definitions contained in clause 3 of ISO/IEC 9646-1 [1], and the following definitions apply:

common part: Part of an ATS written in TTCN, where the declarations of some TTCN objects have been left out, to be provided in a **specific part** of the test specification. The objects left out are objects that are necessary for a complete test specification, but whose values or structures are dependent on the **environment** in which an Implementation under Test (IUT) operates.

environment: Set of all entities within an System under Test (SUT) and their relations to the IUT, affecting the behaviour of the IUT with respect to the protocol(s) under test. Normally the environment is specified by a profile, a part of which the forms IUT.

existing ATS: IS-TTCN ATS that has been developed before the definition of the modularization concept.

NOTE 1: IS refers to the status of TTCN under the International Standard version rather than the Draft International Standard (DIS) version.

functionally equivalent ATSs: ATSs containing the same test cases in the same grouping structure, and assigning the same verdicts to each sequence of events.

modular TTCN: IS-TTCN combined with the features of import and TTCN module.

module interface: List of objects external to the module combined with the list of exported objects.

multi-part ATS: Set of 2 or more partial-ATSs, which together form a semantically complete TTCN ATS.

old ATS: Synonymous for Existing ATS.

partial-ATS: TTCN test specification which is not semantically complete.

semantically complete: TTCN test specification containing at least one test case, one Point of Control and Observation (PCO) type declaration and one Protocol Data Unit (PDU) type definition, and referring only to elements defined/declared in this specification.

source: ATS or Module where testing material is imported from.

specific part: Part of an ATS that is related to a particular **common part**. It defines values or structures of a test specification, which are specific for a given environment in which IUTs may operate, and which are necessary to form a complete TTCN test specification together with the common part. More than one specific part may be defined for a given common part.

testing material: Generic term for documents containing IS-TTCN, independent of whether they are ATSs, Modules or other collections of other objects in TTCN.

TTCN module: Collection of TTCN objects, having a similar structure to a test suite, but not necessarily semantically complete.

NOTE 2: A module can be considered to export a Partial-ATS.

TTCN object: Element of an TTCN test suite or module, having a global identifier throughout the test suite or module, as defined in subclause A.4.2.2 of ISO/IEC 9646-3 [3].

3.2 Abbreviations

For the purposes of this ETR, the following abbreviations apply:

ACSE	Association Control Service Element
ASN.1	Abstract Syntax Notion One
ASP	Abstract Service Primitive
ATC	Abstract Test Case
ATM	Abstract Test Method
ATS	Abstract Test Suite
BNF	Backus Naur Form
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CEN	Comité Européen de Normalisation
CIST	Context Independent Specification Technique
CM	Co-ordination Message
CTR	Common Technical Regulation
CTS	Conformance Testing Services
DIS	Draft International Standard
EG	Expert Group (EWOS term)
ETG	EWOS Technical Guide
ETR	ETSI Technical Report
ETS	European Telecommunication Standard
ETSI	European Telecommunications Standards Institute
EWOS	European Workshop for Open Systems
ExTS	Executable Test Suite
FTAM	File Transfer Access and Management
GSM	Global System for Mobile Communication
ICS	Implementation Conformance Statement
INAP	Intelligent Network Application Part
ISDN	Integrated Services Digital Network
ISP	International Standardised Profile
ITU-T	International Telecommunications Union - Telecommunication Standardisation Sector (formerly CCITT)
IUT	Implementation under Test
IXIT	Implementation eXtra Information for Testing
MAP	Mobile Applications Part
PCO	Point of Control and Observation
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
PIXIT	Protocol Implementation eXtra Information for Testing
PSPDN	Packet Switched Public Data Network
PSTS	Profile-specific Test Specification
PT	Project Team
PTC	Parallel Test Component
PTS	Profile Test Specification
PTS-S	Profile Test Specification Summary
RL	Requirements List
ROSE	Remote Operations Service Element
RTSE	Reliable Transfer Service Element
SCS	System Conformance Statement
SCTR	System Conformance Test Report
SS#7	Signalling System Number 7 (CCITT, later ITU-T)
SUT	System under Test
TCAP	Transaction Capabilities Application Part
TP	Test Purpose
TS	Test Suite
TSS	Test Suite Structure
TSS&TP	Test Suite Structure and Test Purposes
TTCN	Tree and Tabular Combined Notation

4 Problem origin and needs of test specifiers

4.1 The basic problem origin

When test cases have been developed for a protocol embedded in a specific environment, and later test cases for the same protocol in a different environment have to be written, the need arises to re-use the existing test cases as far as possible. The **embedded test method** is defined in ISO/IEC 9646-2 [2]. It is applicable when testing multi-protocol equipment. Control and observation are applied through one or more OSI protocols above the one to be tested (see figure 1).

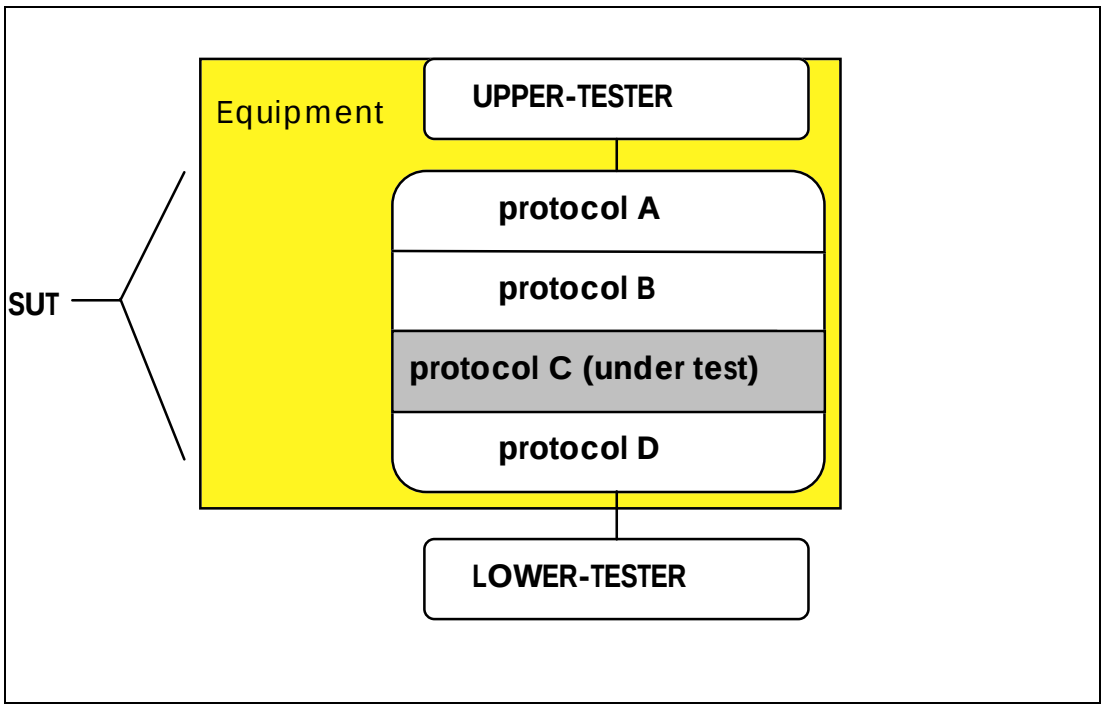


Figure 1: Use of the embedded test method

The need to re-use test cases resulted in a number of activities aimed at writing re-usable test suites. The techniques developed for doing this have in common, that they distinguish between a common part and a specific part. ISO/IEC 9646-2 [2] mentions this possibility to split the test suite in a common and a specific part when using this test method. The common part is dedicated to elements inherent in the protocol under test (here protocol C), while the specific part is dedicated to the environment of the entity implementing the protocol under test within the SUT. Some conformance testing development groups have chosen this option, though compatibility with TTCN as specified in ISO/IEC 9646-3 [3] plus addenda has not been achieved. For more details on this approach, see also clause 5 and/or EWOS/ETG 022 [19].

4.2 Factors complicating the problem

A number of "variants" of embedded testing exist: some typical examples are explained in the following subclauses.

4.2.1 Co-ordination of entities

The relation of the (protocol-) entities in figure 1 is a vertical one: the lower entity provides a service to the higher entity. Horizontal relations of such entities are also possible, as shown in figure 2 below:

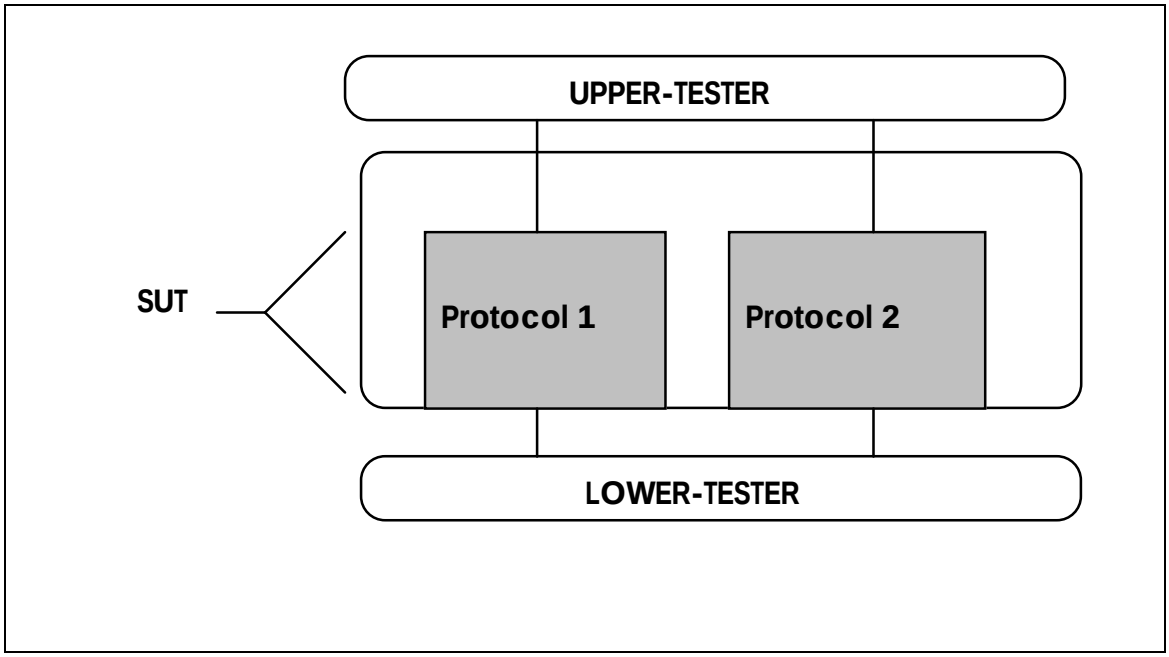


Figure 2: Co-ordinated protocols

The protocols in figure 2 could, for example, be instances of communication at the network layer, that handle connections on subnetworks with a relative independence, but with some co-ordination required. To be more specific: in an Integrated Services Digital Network (ISDN) environment; protocol 1 could be used to provide a circuit-switched or packet-switched connection of a B-channel using the D-channel protocols while protocol 2 is an instance of communication operating on the B-channel.

While operating on the B-channel, the connection on the D-channel has to be kept active. Experience on testing those kinds of related entities has been gained. For example, in the profile test specification for ETS 300 080-1 [15] by taking the "parallel" operation and the re-use of existing test suites into account. The result is contained in ETS 300 080-2 [16]. For a short explanation see also clause 5 below.

4.2.2 Multi-party testing

The multi-party context has been defined in ISO/IEC 9646-2 [2], and has its counterpart in "Concurrent TTCN" as specified in ISO/IEC 9646-3 AM.1 [4]. "Concurrent TTCN" introduces the concept of Main Test Component and Parallel Test Components, which are co-ordinated by the exchange of co-ordination messages. Generally one can say that "Concurrent TTCN" proves its usefulness when in a testing context several instances of communication act rather independently, so that they "can be put" in individual test components.

The concept can, for example, be used in a Signalling System no.7 (SS#7); Global System for Mobile Communication (GSM); or ISDN context. In fact, in the example of the previous subclause, protocol 2 (X.25) is tested in a Main Test Component, while simultaneous D-channel operations are treated in a Parallel Test Component.

4.2.3 Profile testing methodology

A profile is putting requirements on the use of one or more protocols within an SUT. These requirements can be related to a single protocol or to the interaction of two or more protocols. Profile testing is introduced in ISO/IEC 9646-6 [8] and specialised for use in Europe in ETS 300 406 [14]. Profile testing is concerned with the testing of individual protocols of a profile, and may also be concerned with testing the interactions of two or more protocols of the profile.

For the objective of this ETR: giving rules for the re-use of ATSs, profile testing is one essential subject to be addressed.

In fact, all the situations and problems described in the previous subclauses occurred in, or are applicable to, profile testing. Many profiles have been defined and will be defined, but also many of them will share

some protocols (possibly with somewhat different requirements on optional protocol elements). For example, X.25 may be used on a dedicated access to a Packet Switched Public Data Network (PSPDN) or in an ISDN profile; X.400 may be used in a stationary GSM equipment or in a mobile GSM equipment; many ISDN profiles will share the use of the D-channel protocol at layer 2 (or also on layer 3).

For testing equipment implementing such profiles it seems natural to re-use "testing material" developed to test a particular protocol, or even developed for profile-specific purposes. This ETR expresses some rules applicable to such a re-use in clause 7.

4.3 What should be achieved

Summarising the needs of test specifiers, the following can be stated:

a) TTCN itself does not support sufficiently the re-usability of testing material. An extension should be defined;

b) application rules for the use of (extended) TTCN dedicated to the re-use problem should be given;

c) a scheme for making testing material visible and available for test suite writers should be provided.

The features available for the re-use of ATSs and the rules for their application should have the following general properties:

1) Generality
The features should be so general that ad-hoc solutions can be avoided. The applicability should be guaranteed but not limited to the embedded test method, profile testing and "Concurrent TTCN".

2) Compatibility
The proposed extensions to TTCN should be additions in the sense that existing ATSs in TTCN are compatible to the extended form.

3) Simplicity
The proposed extensions and their application rules should continue as intuitively as possible what people are currently doing in this area. Complicated constructions should not be necessary to follow the scheme.

4) Modularity
The proposed scheme should support modularity in the description of how to re-use testing material. This is necessary to have readable and also well maintainable test specifications.

5) Tool-applicability
The proposed scheme should be applicable by test tools. The transition to the new scheme should not be too expensive for tool providers.

6) Maintainability
The proposed features for the re-use of ATSs should fit in the maintenance and classification schemes of existing bodies involved in conformance testing, and also be open for extensions in the future.

5 History of solution attempts

The problems described in the previous clause emerged, in some form, in many conformance testing projects developed for protocols used in multiple embeddings. In most cases the problem was dealt with in an ad-hoc way, but in the CTS-2 XMHS and FTAM projects more structured solutions were developed. Later, the CTS-3 NM (CMIP and CMISE testing) project produced a more flexible solution, based on the CTS-2 experiences.

At the same time, other projects, especially those dealing with lower-layer (also ISDN D-channel) protocols still proceed in an ad-hoc way. One reason for this is that test configurations and other aspects of these protocols are too specific to develop a standard method. So each time specific, one-time solutions were chosen.

The remainder of this clause describes the attempts made so far.

5.1 CTS-2 projects: common and specific parts

The European Commission initiated the Conformance Testing Services (CTS) program. This program aims at the establishment of harmonised testing services for Information Technology and Telecommunication products in support of a European certification scheme. In the CTS-2 XMHS and FTAM projects it was recognised that a number of ATSs should be written in such a way that re-use of the ATSs (ATS) would be possible when testing IUTs for X.400, X.500 and FTAM.

The projects decided that the ATSs for the supporting protocols of these application layer protocols had to be written in such a way that re-use was possible. It concerned the protocols for Remote Operations Service Element (ROSE), Association Control Service Element (ACSE), Presentation layer, and Session layer.

In order to be able to write these ATSs, test suite production techniques supporting re-use had to be developed. It happened that for each of the protocols a different technique adapted to and named after the specific protocol was developed. The common feature of each technique was to define an ATS in two parts, a **common part** and a **specific part** which are merged later. The common part contains the core of the ATS, with behaviour tables, and the specific part contains the elements that are "embedding-dependent" (dependent of X.400, X.500 and FTAM). For each embedding one specific part exists. See figure 3.

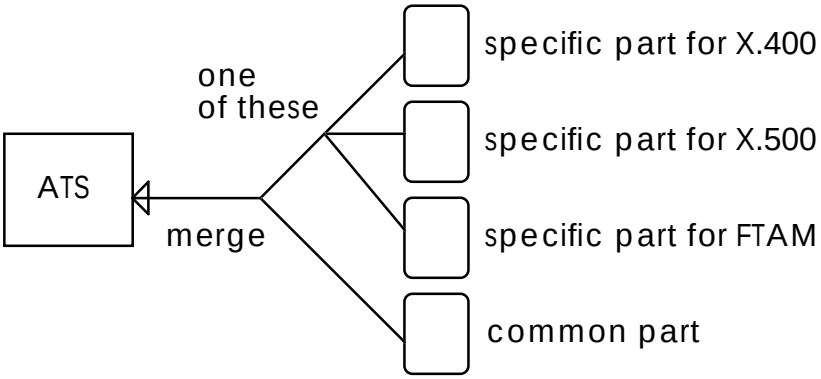


Figure 3: Combining common part and specific parts

The techniques each make a different split "common/specific". For a detailed discussion see EWOS/ETG 022 [19]. A short overview of each technique is given below:

ROSE-technique: This technique uses parameterisation of "embedding-dependent" constraint fields. The specific part then consists of a "Test Suite Constants" section defining the values for the actual parameters. The constants in the specific part are referred to, but not declared, in the common part.

ACSE-technique: This technique uses a complicated method in which base constraints are defined in the specific part and modified using a REPLACE statement in both parts.

presentation-technique: This technique uses a method similar to the one used in the ACSE-technique, but base constraints are defined in the common part here.

session-technique (also called **abstract model-technique**)**:** This technique uses User Defined Operations to take care of analysing the incoming, embedding-dependent PDUs and producing correct responses.

None of these techniques is general enough to use all types of different protocols, so it can be concluded that the problem of arranging the interaction between the common part and the specific part has been recognised, but has not been solved adequately.

5.2 The CTS-3 NM project: advanced common and specific parts

In the CTS-3 NM project (CMISE and CMIP testing), also test suites for supporting protocols had to be developed. Because this project wanted to build upon earlier experiences it looked at the results of the previously mentioned CTS-2 projects.

The project evaluated all techniques developed in the CTS-2 projects, and came to the conclusion that there were two major drawbacks to them:

a) **interaction problems between common and specific parts:** Where to specify constraints, how to transfer them between parts. These problems were not adequately solved, the techniques were not flexible enough.

b) **embedding specific test steps:** It was recognised that some parts of the behaviour trees depended on the embedding. A preamble, for example, may consist of an embedding independent part and an embedding dependent part. The first is used to reach a specific state in the IUT and the second is used to reach a desired state in the embedding layers. So, the embedding dependent part of test steps should be incorporated into the specific part as well. This problem was not even addressed in the CTS-2 projects.

So, the project developed a more generic technique called CIST (Context Independent Specification Technique). This technique solved the problems and contains good concepts to be used in the definition of a truly universal technique.

5.3 ETS 300 080 profile testing: modify operations on a test suite

ETSs 300 080-2 [16] to ETS 300 080-4 [18] will provide test specifications for lower layer protocols for ISDN terminals. These test specifications are the first profile test specifications to be provided by ETSI. The profile is specified in ETS 300 080-1 [15]. It covers the ISDN D-channel protocols, X.25 protocols on layers 2 and 3 of the B-channel and the T.70/ISO 8073 [20] protocol on the transport layer. The most interesting point for this ETR is contained in the specification of the use of the X.25 layer 3 ATS contained in ISO/IEC 8882-3 [10] for the purposes of this profile. The following kinds of modifications of the internationally standardised ATSs have been defined in the profile-specific test specification:

- deletion of test cases;
- addition of test cases;
- new constraint declarations;
- new selection expressions;
- replacement of dynamic behaviour of test steps;
- replacement of individual behaviour lines;
- introduction of default behaviour;
- introduction of parallel test components with the X.25 ATS as Main Test Component.

In this profile, the entities above the network layer are either a transport class 0 entity, or a syntax-based videotex application layer (!) entity. The treatment of user data of X.25 packets is not changed in the Profile Test Specification (PTS) with respect to the base ATS. Test suite parameters associated to Protocol Implementation eXtra information for Testing (PIXIT) items are used.

The main objective of the modifications is the adaptation to the ISDN environment, and not to the protocols at the upper boundary of the IUT. In this sense the experiences gained here are complementary to those of the aforementioned CTS projects using common/specific parts.

Since a formal notation for modifications was not available, all the modifications have been expressed verbally, or by the use of TTCN tables (including extensions/parallel test components).

5.4 Ad-hoc solutions of other projects

Other conformance testing project teams have provided ATS specifications, or are currently providing them. With respect to the re-use of testing material, they do not go beyond editorial copying from other ATSs. User data to be transmitted/received in PDUs of the protocol under test are sometimes handled via test suite parameters associated to PIXIT items.

5.5 Statements on re-use in conformance testing documents

A number of documents in the conformance testing standardisation area have addressed the problem of re-use. The statements in each of these documents are summarised in this subclause, in order to come to a complete overview of the current thinking in this area.

5.5.1 ISO/IEC-9646

In ISO/IEC 9646-1 [1], the concept of Layer Testing has changed to the concept of Protocol Testing, in order to be able to incorporate the previously mentioned developments: ROSE, ACSE, and other components are only part of a single OSI Layer, but can now be separately tested.

In ISO/IEC 9646-2 [2] (including draft amendments), the concepts "common part" and "specific part" are introduced. The relevant sections are repeated here:

> **13.1 General:** An ATS for an embedded test method may be specified in two parts: one which is independent of the protocol(s) under which the embedding is to be done, and one which is specific to the embedding protocol(s). The former is called the common part and the latter the specific part.
>
> **13.2.2 Use of TTCN:** For embedded test methods, when the ATS in TTCN is divided into common and specific parts, the specific part shall only contain declarations and/or test step definitions for all unresolved constraint and test step identifiers used in the common part.
>
> **13.2.3A:** When specifying test cases for the embedded method, this ATS specifier may find that parameterising an ATS will allow it to be used, unchanged, under several higher protocols. In this situation the ATS will be a multipart document. The first part will contain the parameterisable test case description. Subsequent parts will contain "actual parameters" for each higher protocol. To have a complete ATS description it is necessary to have both the part containing the parameterisable description and one of the protocol specific parts.

5.5.2 ETS 300 406 (European conformance testing methodology)

ETS 300 406 [14] provides rules and guidance on how to develop conformance testing standards within ETSI. The relevant clauses are:

ETS 300 406 [14], subclause 5.8, Recommendation 2:

> *".... Re-use of test cases should not be prevented by the definition of ATS in a profile-specific context ... "*

ETS 300 406 [14], subclause 5.7, rationale for Recommendation 1, item c):

> *"... In the context of a profile, assumptions can be made on the other parts of a system, e.g. adjacent layers.... Defining tests for a base specification without knowing anything on the other parts of a System Under Test often leads to either defining tests incompletely (e.g. without specifying what is contained in the data part of the PDUs sent), or tests which are totally inapplicable..."*

Here one can conclude that for each protocol for which tests are to be developed, first it has to be established in which "embedding" the protocol will likely be found. Then tests for the protocol under that embedding can be made. However, it would be more convenient if the tests were embedding-independent.

In ETS 300 406 [14], subclause 7.1, on Abstract Test Methods (ATMs) used for functional subsets of a protocol, it is mentioned that sometimes more than one test suite, each using a different ATM, must be used for testing (however this is not recommended).

In ETS 300 406 [14], subclause 10.1.6, guidance is given on the way to deal with ATS that have components in common. Two examples are given: one mentioning Embedded ATSs for the common stack of upper layer protocols (ROSE, ACSE, Presentation, Session) and one mentioning the test suites developed for ISDN D-channel layer 3 (Basic Call Control). The problems with this approach are then discussed:

Quote "....

NOTE: The problem of partial or modular test suites is threefold:

a) TTCN test components are not independent Modules in a TTCN test suite, and the components belonging to the four parts of the suite: Overview, Declarations, Constraints, Behaviour have cross-connections between each other. TTCN should be enhanced to support "extraction" of components.

b) A precise referencing scheme, analogous to the PTS-Summary, would be needed to express that a test suite is composed of several (common or specific) components, specified in several standards.

c) Maintenance of test suites has to be aligned with a modular approach. In fact, the modification of a component may affect several test suites.

..." unquote.

Also, in ETS 300 406 [14], a general rule is stated that has to be followed as long as there is no official methodology for this:

> *"... When different test suites have some components in common (test cases, test steps, constraints, etc.) the components in question shall be duplicated, and each test suite shall remain complete, self-standing and independent. ..."*

5.5.3 BC-IT-226 phase 1 task D/14

The objective of BC-IT-226 (CEN/ECITC/PT001) phase 1, Task D/14, was the definition of a set of topics, and a plan for their resolution, to address outstanding methodology issues in OSI conformance testing. About thirty-five topics were defined. Three of them are of relevance to the partial and multi-part test suite subject:

Modular test suites for the application layer (topic 23): this topic mentions re-use of test cases for ROSE and ACSE, in various application contexts;

Common embedded testing under application layer (topic 31): this topic extends from topic 23 and mentions the concept of common/specific parts. It states that techniques to deal with the common/specific parts need further study and that there are implications on TTCN and TTCN tools;

Partial-TTCN Test Suites (topic 45): this topic generalises the subject and states the need for a standard approach on modular re-usable test suites.

All three topics are covered in this ETR.

Task D/14 mentions, as example of members of the Test Specification Library, the embedded test suites for ACSE, Presentation and Session.

5.5.4 ETR 141 - The TTCN style guide

ETR 141 [11] doesn't give guidelines on the specification of common or specific parts, but a reference is made to the EWOS/ETG 022 [19]. Some of the rules of the style guide are picked up in clause 7 of this ETR, where additional rules are given.

5.6 Modular TTCN

This is the key word for still ongoing attempts to extend ISO/IEC 9646-3 [3] with the concepts of import and TTCN Modules. There is no doubt among experts that such features will be included in ISO/IEC 9646-3 [3], but there are still some open issues concerning the exact realisation of these concepts (see annex C).

5.7 Modularity on the tool level

Some kind of modularity is currently supported internally by some editing tools. Two have been studied and are presented here: ITEX and Concerto.

5.7.1 Modularity support from the ITEX TTCN editor

The ITEX TTCN editor supports modularity to a certain level.

First, in order to support modularity, one has to bear in mind that support of modularity must allow use of semantically incomplete ATSs, where a semantically incorrect ATS is an ATS without any PCOs, no test cases, no constraint part etc. A semantically correct ATS is thereafter created from a collection of several (at least two in order to achieve modularity!) semantically incorrect ATSs.

The ITEX TTCN editor supports modularity in two different ways:

a) edition of fragment;
b) edition of self-containing ATSs.

A TTCN fragment is a piece of TTCN which is syntactically correct, and not smaller than a table. Examples of fragments are a Simple Type Definition, a TTCN Abstract Service Primitive (ASP) Constraint, a Test Case etc. An ATS containing TTCN fragment(s) is never semantically correct.

A self-contained ATS contains everything needed in order to verify its correctness. So, a Simple Type Definition can be self-contained if it is defined in terms of TTCN pre-defined types and at the same time be a TTCN fragment. The TTCN ASP Constraint mentioned above can never be a self-contained ATS as it needs its corresponding type in order to verify (analyse) the correctness of the TTCN ASP Constraint.

The ITEX TTCN editor provides a tool, the "Merge" tool, for joining two TTCN fragments and/or self-contained ATSs into one ATS that is either a collection of TTCN fragments or self-contained. By repeating this procedure, a semantically valid ATS is achieved.

5.7.2 Modularity support from the Concerto/TTCN editor

The Concerto/TTCN editing tool allows the breaking down of ATS into sharable fragments. From an editing point of view, a test suite, or a fragment, can be composed of fragments. A fragment is a piece of TTCN which is syntactically correct, and not smaller than a table such as a **Declarations Part**, an **ASP Constraint Declarations** subpart, or a **Test Step**. The fragmentation remains internal to the tool: it is not visible in the Machine Processable (MP) files produced from a fragmented suite, neither on the Graphical Representation (GR) printings. A fragment can be shared among several test suites or other fragments and functions are provided to manage this fragmentation. Fragments can be managed as standard objects in Concerto and particularly be versionned independently of the test suites that include them.

This functionality of the tool is used to produce and manage ISDN layer 3 conformance test suites. Test suites to test the Basic Access and test suites to test the Primary Rate Access have a majority of test cases in common. The only test cases that are specific to each ATS, and thus not shared, concern the suspend/resume mechanism. In the same way, the test steps used as preambles are specific to each suite.

6 General scheme of re-use

6.1 Introduction

The general aim of this ETR is to describe rules for the re-use of ATSs in different contexts.

Clause 5 described the problems encountered with such re-use and some solution attempts. The known solutions are sufficient for the particular needs of their developers, but not general enough for the conformance testing methodology developed in ISO/IEC 9646 [1] to [9] and as specialised in ETS 300 406 [14] for Europe.

This clause presents a general scheme of entities and procedures considered to support an efficient re-use of testing material. Part of this scheme already exists, also, other parts are shown which could be

installed in the future. Regarding future parts, this ETR indicates possibilities. The decision on implementing or using such possibilities is outside the scope of the ETR.

Writers of ATSs normally take the protocol(s) under test, the Protocol Implementation Conformance Statement (PICS), the Test Suite Structure and Test Purposes (TSS&TP) and the Test Method as a starting point for their development. Besides the consideration of these essential documents, the question of **which** related testing material can be found **where** is interesting.

Subclause 6.2 below deals with this question. The availability of ATSs is particularly important for writers of profile test specifications, where the problem typically is how to use existing ATSs in the context of the profile.

Subclause 6.3 gives an overview of the possibilities **how** to re-use existing testing material. This is refined in subclause 6.4 for the case of editorial copying of testing material, and in subclause 6.5 for the case of formal import of Modules.

Due to the general nature of this clause, no rules are given here, but, when applicable, it is indicated which subclause under clause 7 gives rules, examples, or more information on the related kind of re-use.

Subclause 6.6 describes the general context of applicability of the rules contained in this ETR with respect to the International and the European conformance testing methodology and to different types of IUTs.

6.2 Availability of testing material

When the first Abstract Test Specifications using TTCN were written, they were available in the groups creating them (e.g. CTS-2, etc.), and were often passed to experts asking for them in an informal way, before they were standardised. In the meantime, liaisons between standardization organizations (such as ETSI) and groups creating test suites (e.g. CTS) have been improved and formalised. But still you can hear experts ask: "is there a list of test suites .. for protocol(s) ... in organization ...?"

Figure 4 below shows possibilities of how testing material can be kept and made available (but does not claim completeness):

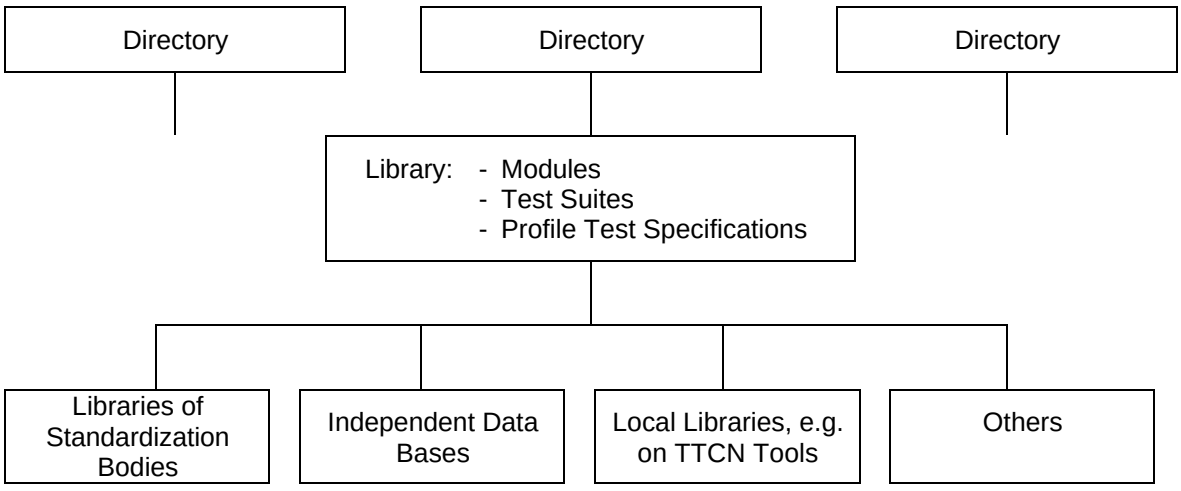


Figure 4: Testing material available for re-use

The indicated directories could be listing libraries, or could be directories inside libraries listing testing material, and mixing is also possible.

The contents of such libraries: Modules, Test Suites and Profile Test Specification, has been selected mainly from the standardization point of view. Libraries local to a company, or a tool, could of course also contain test specification fragments which are not Modules in the sense of Modular TTCN.

NOTE: The question according to which criteria Modules should be constructed or isolated from existing testing material before entering a library, is discussed in subclause 7.3.

Standardization bodies maintain their own libraries and indexes. Commercial data bases with testing specifications can also be expected in the future (if they do not exist yet). Local libraries on testing tools do already exist, but their contents, structures, and usage are outside the scope of this ETR.

Further considerations on libraries and tools are contained in clause 8.

6.3 Different ways of re-using testing material

When the writer of an ATS or a profile test specification has identified existing testing material that could be re-usable for his purposes, he still has several different possibilities of **how** to re-use it.

Figure 5 gives an overview on different ways in which existing testing material could be re-used. The numbers indicated in the boxes indicate subsequent subclauses of this ETR, in which further explanation can be found.

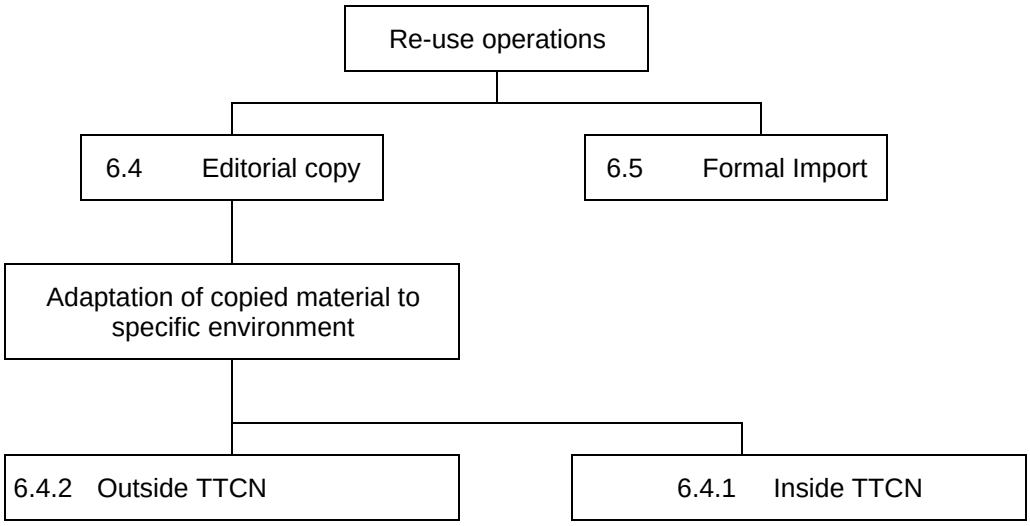


Figure 5: Ways of re-using testing material

The point of view taken so far in this subclause is that some material has been identified for re-use and is used in some way. But there is also the reverse problem that the writer of testing material should provide it in a way suitable for re-use.

Though writers of ATSs generally cannot foresee all environments in which IUTs having implemented the protocol(s) under test may operate, there are some points, where elements of an ATS are likely to be touched by a particular environment, and where ATSs can be written in a way that makes them as insensitive as possible in this respect. This covers the modularization aspect, as well as the way of writing certain elements of an ATS. Rules and examples on this subject are contained in subclauses 7.2 and 7.3.

6.4 Editorial copy of testing material

This is the main procedure that has been used until now when re-using testing material, whose result is to be standardised. Except for the ASN.1 Type Definitions by Reference, IS-TTCN as defined in ISO/IEC 9646-3 [3] and ISO/IEC 9646-3 AM.1 [4] currently does not provide any other means. According to ETS 300 406 [14], this is the only allowed procedure when creating Abstract Test Specifications.

NOTE: This does of course not prevent from modularization of testing material on the tool level (or more generally in libraries not claiming to contain only standardised testing material).

6.4.1 Adaptation of testing material remaining inside TTCN

The adaptation of testing material remaining inside TTCN is restricted by the fact that modification operations are not defined inside TTCN. The writer has to use editorial operations to make the copied material fit to the particular needs of the current test specification. The production process is neither recorded nor standardised, only the result will be standardizable, independent of which sources were used.

6.4.2 Adaptation of testing material using features outside TTCN

This is related to the possible use of something like "pre-processing". Pre-processing can, for example, occur on the tool level, using local libraries; Modules; and ATSs, and features to link them. For this way of pre-processing the same qualification applies like in the previous subclause: only the result is standardizable; the way in which tools operate is outside the scope of this ETR.

However, pre-processing is also applicable to the abstract specification level. This possibility mainly applies to profile testing, where one will as far as possible **use** existing ATSs for testing the single protocols of the profile (avoiding having to write completely new ATSs for a protocol), and is less applicable to the writing of new ATSs (which have to be standardised on their own, and where the pre-processing could be hidden in the production process).

To be more explicit:

When writing a profile test specification, the following things typically occur when making use of a particular ATS for a particular protocol in the profile:

a) not all test cases are applicable to the profile. Deselection of those test cases can (mostly) be done using the selection expressions inside the ATS;

b) new test cases have to be defined (profile-specific, or protocol-specific but not covered in the ATS). This normally implies that additional constraints etc. have to be defined;

If case b) occurs, the resulting profile-specific ATS is not as a whole specified (only) in TTCN. But even worse things can occur:

c) objects used to a broad extent in the ATS, e.g. preambles or defaults or user data in constraints, are not adapted to the environment given by the profile (see, for example, parts 2 to 4 of the revised version of draft (STC) ETS 300 080 [16], [17] and [18]).

If cases b) and/or c) occur, two possibilities exist:

1) duplicate the ATS and attach the complete and modified new ATS as a Test Specification to the Profile Test Specification;

2) allow to specify modifications in some way.

Basically two possibilities exist for the latter case:

- allow textual explanations, e.g. "replace the default references of test cases ... by NewDefault", or "include the constraints declarations of subclause x.y.z into the ATS";

- use some formal way of specifying modifications of ATSs (or Modules).

NOTE: At the moment only the first possibility is being considered.

6.5 Formal import of testing material

Except for the ASN.1 "Type Definitions by Reference", the importing is a new feature. As mentioned earlier in this ETR, there are still ongoing attempts to extend ISO/IEC 9646-3 [3] with the concepts of import (and TTCN Modules). For some open issues concerning the exact realisation of these concepts see annex C.

When using the feature of importing, the ATS writer should carefully select **what** to import and **what not** to import. Only material that fits exactly (except for renaming) should be imported. All other material must be defined explicitly, where of course editorial copying/modification may be used for the explicit declarations in the phase of writing the ATS or the module.

Using this feature (when it is applicable, i.e. when no modification of the imported material is necessary) will reduce the amount of testing material that has to be editorially copied and also benefit from the re-use of material that has been checked and verified in other situations.

From all the outstanding questions and problems with the concept of Modules in TTCN, two basic views on that concept that have been expressed by experts so far (see also annex C):

View 1:

A module is a Partial-ATS, that contains all objects it refers to either explicitly, or by inclusion of, some other module. All TTCN parts (Declarations Part etc.) are made optional. The requirement that at least one PDU declaration and one test case shall be contained in a TTCN Test Specification is removed.

View 2:

A module is a Partial-ATS, that need not contain all objects it refers to (not even by reference to some other module). Again all TTCN parts (Declarations Part etc.) are made optional.

View 1 is more restrictive than View 2, but it would be possible to retain more syntax and semantics checks on the module level.

EXAMPLE: The common and specific parts in CTS 3 NM would not be Modules in the sense of View 1, since the specific part contains the Declaration of test suite Constants the common part refers to.

View 2 is more flexible, but nearly no semantics checks would be possible on the level of a single module; only when Modules are combined to form a semantically complete IS-TTCN ATS, the verification of semantics could be realised.

It is currently not decidable, which view will survive. The rules for the construction and use of Modules contained in the subsequent clauses will therefore be expressed with some assumptions on the applicability.

6.6 Applicability of the concept with respect to types of IUTs

The Testing Methodology of ISO/IEC 9646 [1] to [9] and the European Testing Methodology ETS 300 406 [14] still leave several parameters open giving information about which "types" of IUTs are tested in which way. In order to state the applicability of the rules given in this ETR, the basic test configuration of ISO/IEC 9646-1 [1] is repeated in figure 6:

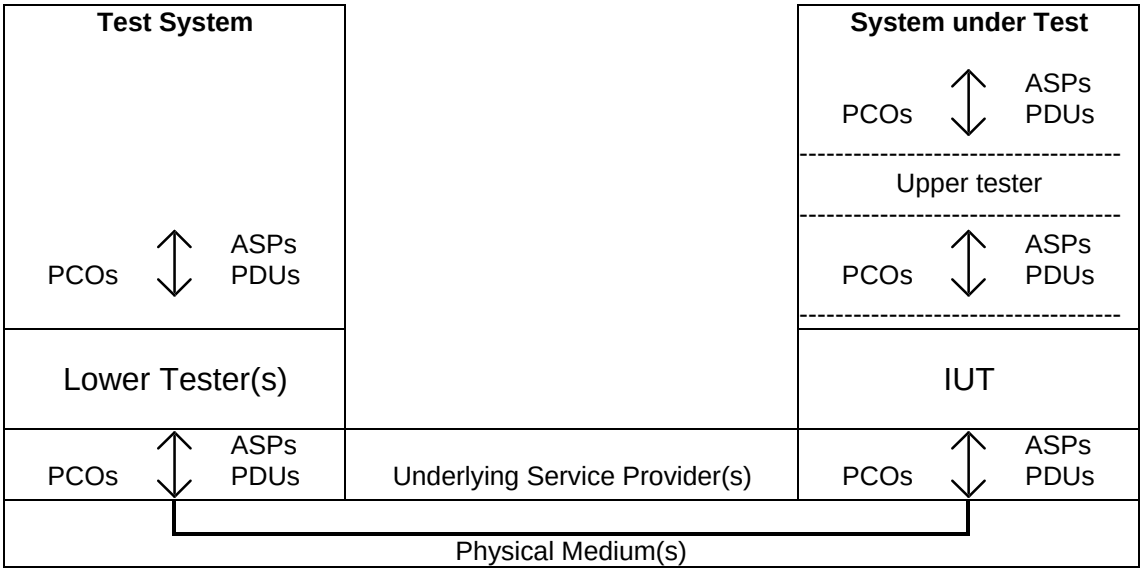


Figure 6: General test configuration

Figure 6 shows the embedding of the IUT in the SUT and the possible relations to one or more components of the testing system. Other entities in the SUT that could have a relation to the IUT are not explicitly shown. Such entities could reside at the upper boundary of the IUT (the IUT providing service(s)); at the lower boundary (providing service(s) to the IUT); or "at the same level" (e.g. when the

IUT is a network layer entity requiring co-ordination with another network layer entity currently not being under test (ISDN B/D-channel co-ordination)).

It is this general embedding of an IUT in the SUT that can put conditions on the way an existing ATS can be used to test the IUT.

The problem is **not** restricted to situations where the embedded test method is applied. The difference between the embedded test methods and their not-embedded forms is, that in the embedded form the control of the IUT while testing is not directly at the upper service boundary of the IUT, but through some other entities using the service of the IUT. But, independently of whether the control is direct or not: the context-dependent **contents** and **sequence/order** of service data units exchanged at the upper boundary of the IUT may have an impact on the use of an ATS to test the IUT. It is also not only the environment of the IUT at the upper boundary, that can have such impacts. In the clause 7 more detailed investigations will be made on this subject.

NOTE: In this ETR relations between an IUT and its environment are stated in terms of services, PDUs and co-ordination operations. Particularly the terms "embedding protocol" or "embedded under" are not used further.

General Statement of applicability:

The guidance and the rules given in this ETR are intended to be principally applicable to all variations possible in the testing methodology as specified in ISO/IEC 9646 [1] to [9], and in ETS 300 406 [14]. Particularly, they are not restricted to:

a) either higher or lower OSI layers;
b) particular test methods, embedded or non-embedded form;
c) single party or multi-party context.

7 Rules for re-use

7.1 General

The proposed rules are based on the current status of development of ISO/IEC 9646-3 [3], where ISO is likely to include Modular TTCN. Aspects of Modular TTCN are mainly discussed in subclauses 7.3 to 7.7, while subclause 7.2 is more general. Subclause 7.3 takes up the problem of constructing Modules, i.e. of "distributing" testing material on Modules. Subclause 7.4 starts from the converse point of view: to construct test suites from given Modules or test suites. In subclause 7.5 the problem of re-use of existing test suites not written in a modular way is treated, while subclause 7.6 takes up some ad-hoc solutions of the past for combining common/specific part test suites (as described in clause 5), and discusses the relation of these solutions or "techniques" to Modular TTCN. Subclause 7.7 shows how Modular TTCN could be used in profile testing.

When open issues of Modular TTCN are touched (see annex C), rules and examples are expressed conditionally.

The status of the material available for re-use is the following:

a) There is a pool of TTCN Modules, at the moment empty, which is gradually filled-up when ATS writers divide their ATSs into Modules. ATS writers may want to use material from this pool.

b) There is a pool of existing ATSs from which ATS writers may want to use material.

This situation generates the following questions that need to be answered in this clause:

1) What are the re-usability aspects of ATSs? (subclause 7.2).

2) How does Modular TTCN work? (subclause 7.3).

3) Which rules for modularization of test suites are useful? (subclause 7.3).

4) How to create new ATSs with the use of Modules? (subclause 7.4). Which Modules to use, which Modules to create? How to merge them? What changes need to be made to the imported ATSs or Modules?

5) How to use and modularise existing ATSs, how to apply rules for modularising? (subclause 7.5).

6) How to deal with "existing Modules" (e.g. ROSE, ACSE common and specific part)? (subclause 7.6).

7.2 General rules for re-usability of test suites

7.2.1 Context of applicability

This subclause is applicable in the case where it is the intention to make test material to be developed suitable for re-use as defined in this ETR.

The rules are general in the sense that they do not specifically refer to Modular TTCN or other re-use techniques: they are meant to serve "good test suite writing practice". If used, the rules result in test suites that are, in a general way, easier to re-use (although possibly more difficult to read), and simpler to modify. If not, (as is the case with some existing test suites) a test suite can only be re-used through a large modification action. This implies that the designer of the re-usable test suite should be aware of the modification possibilities in a test suite. Only then can he/she write a truly re-usable test suite.

7.2.2 Re-use problem areas

In order to obtain a sight on all implications re-usability has on his actual work, the developer of a new ATS can use the approach given in the remainder of this subclause.

As many problems of re-usability of an ATS come from a modification of the environment, the basic idea is to investigate, for each interface of the IUT within the SUT, which modifications can occur. An IUT mainly has three classes of interfaces: the lower interface; the upper interface; and sometimes an interface with another "plane" at the same layer. Additionally, in the IUT implemented protocol itself can be subject to change inherent to the context of use. For each of these items, questions to ask and examples are listed.

a) Effect of the environment at the LOWER boundary of the IUT.

1) Is it likely that the services provided by the lower layer(s) to the IUT may change?

   EXAMPLE 1: Using Basic Access or Primary Rate Access in ISDN

   The ISDN layer 2 D-channel protocol can operate over i) Basic Access, ii) Primary Rate Access. Using Basic access, layer 1 can be activated or deactivated, while in Primary Rate Access this is not possible.

2) Can it happen that the same service is provided, but by a different protocol?

   EXAMPLE 2: Same service provided by different protocols

   i) X.25 layer 3 protocol can operate over X.25 layer 2 or over the ISDN D-channel protocol at layer 2.

   ii) X.400 can operate over Reliable Transfer Service Element (RTSE) or over ROSE

3) If the protocol at the lower boundary needs some (or no) initialisation, is it likely that modifications may appear in this procedure?

   see EXAMPLE 1

4) Is the protocol at the lower boundary connectionless, connection oriented, or both? Does this option make difference in the initialisation procedure?

b) Effect of the environment at the UPPER boundary of the IUT.

1) Is it likely that the protocol under test provides a different service depending on which protocol(s) are implemented above the upper boundary?

EXAMPLE 3: X.25 protocol providing different services

X.25 can provide the OSI Network Service (without Q-bit use), or provide the Bearer Independent Service for videotext, which may also use Q-bit procedures.

2) Is it likely that the protocol implemented in the IUT provides a given service for possibly different protocol(s) implemented above?

EXAMPLE 4: Upper layer protocols providing service to different protocols.

i) ROSE offering service to X.400, X.500, and other application layer protocols. ROSE is designed for use by many application layer protocols, so a test suite written for ROSE may be re-used many times.

ii) ACSE offering service to X.400, X.500, FTAM. ACSE is, as ROSE, defined for use by many application layer protocols.

iii) Presentation layer offering service to many application layer protocols.

iv) Session layer offering service to all kinds of protocols.

3) Will the same services be provided in a different way (i.e. with a different sequence of ASPs/PDUs) by the IUT depending on the upper environment? Does the protocol give the choice between connectionless and connection-oriented and does this choice influence the sequence and the contents of data PDUs?

EXAMPLE 5: Direction dependencies in SCCP connection oriented protocol.

The GSM A interface (between BSS and MSC) uses the connection oriented SCCP protocol, but the higher layer BSSAP is not symmetrical; the order and contents of data PDUs sent in an SCCP connection depend on the "direction".

4) Will a PCO always be accessible at the upper layer boundary? If not, which combinations of upper layers are possible?

EXAMPLE 6: Embedding of protocols in SS#7 stacks

The SCCP upper boundary is not often accessible in a real SS#7 environment and the upper layer can be constituted by a Transaction Capabilities Application Part (TCAP) + Mobile Applications Part (MAP) + GSM-Application or by a TCAP + Intelligent Network Application Part (INAP) + IN application.

c) Effect of the environment at the same level as the IUT.

1) Does the IUT act as a control layer for another "plane"?

EXAMPLE 7: ISDN D-channel controlling the usage of the B channel

2) Can another "plane" interact directly (i.e. by a control interface on the IUT behaviour)?

EXAMPLE 8: Direct interaction

i) RTSE using ROSE service (at same level).

ii) A network management interaction on SCCP.

d) Effect of profiles and technical contexts on the IUT.

1) Is the protocol/profile specification subject to evolution (i.e. is there a planning of evolution)?

   EXAMPLE 9: SCCP versions (1988 and later 1992), GSM MAP phase1/phase2

2) Do multiple profiles exist for the specification?

   EXAMPLE 10: For X.400 a large quantity of profiles exist.

3) Has the protocol a broad field of applicability in the technical context?

   EXAMPLE 11: Many occurrences of the X.25 packet layer protocol in profiles.

When all these questions have been answered, the developer has a good overview on which parts of the ATS are sensitive to re-use. Then, care should be taken during the development of these particular parts. In the following subclauses, rules for writing re-usable test material will be established, which together cover the topics mentioned. For each rule, applicable examples are mentioned.

7.2.3 Rules for writing re-usable test material

7.2.3.1 Principles

Before defining specific rules, some "principles" which are always applicable can be expressed:

Principle 1: Relation of test body to the environment

Test cases should be written such that the **test body** contains only valid sequences of events with respect to the protocol under test, and with no explicit relation to the environment. Behaviour related to environment should be "hidden" in preambles, postambles, other test steps and possibly in parallel test components.

Principle 2: Rules from the TTCN style guide

The rules of the TTCN style guide - ETR 141 [11] should be followed. Many of them lead to more re-usable ATSs.

When applicable, the rules in the following table 1 from the ETR 141 [11] TTCN style guide are quoted (some of the rules are rephrased for brevity and for applicability to the re-use subject):

Table 1: Rules from the TTCN style guide

No.	Rule
12c	Map the applicable ASPs/PDUs from the service/protocol standard to ASP/PDU Type Definitions in the ATS.
12d	Map the structure of ASPs/PDUs in the service/protocol standard to a corresponding structure in the ATS.
14a	Develop a design concept for the complete constraints part, with respect to constraint parameterisation.
15a	Design the relation between base constraints and modified constraints always in connection with parameterisation of constraints.
16b	When the ATS is used in different profile environments and the types and values of embedded PDUs cannot be predicted, Dynamic Chaining (constraint parameterisation) is normally the choice.
17b	Keep the number of formal parameters small.
18b	Use test suite constants instead of literals, when appropriate.
23	Parameterise the constraints used within a test step.
28	Use parameterised test steps to ensure re-use of test steps within test cases for different needs.

7.2.3.2 Sample test case

The following test case **TC_Template** is used as a sample, which will be referred to below, when explaining specific rules:

TC_Template

Test Case Name: TC_Template **Group:** **Purpose:** **Default:** Def_Template **Comments:**					
Nr	**Label**	**Behaviour description**	**Constraints ref.**	**Verdict**	**Comments**
1 2 3 n n+1 n+2	start body pas_lb pa1	+Tpl_Preamble LT!Event_one . . LT?Pass_branch_last_event +Tpl_Postamble	 c1 (parm_1) c2 (parm_2)	 (P)	

Definition of Tpl_Preamble:

Nr	**Label**	**Behaviour description**	**Constraints ref.**	**Verdict**	**Comments**
1 2	init p_body	+Tpl_Test_Initialisation LT! First_Preamble_Event			

Definition of Tpl_Postamble:

Nr	**Label**	**Behaviour description**	**Constraints ref.**	**Verdict**	**Comments**
 m m+1	 last_evt exit	... LT! Last_Postamble_Event +Tpl_Test_Exit		 (P)	

Definition of Tpl_Default:

Nr	**Label**	**Behaviour description**	**Constraints ref.**	**Verdict**	**Comments**
	exit	LT? Tpl_Fail_Event CANCEL +Tpl_Test_Exit ?TIMEOUT CANCEL +Tpl_Test_Exit ?OTHERWISE CANCEL +Tpl_Test_Exit		(F) (F) (F)	

7.2.3.3 Test case initialisation rules

Rule 1: Test step in preamble

A test case should contain at least one test step in its preamble for initialisation related to the environment.

In order to handle any kind of initialisation of the test case when the test case is going to applied in a different environment, it is recommended to provide a test step inside each preamble. This test step should be devoted to initialising environmental aspects, not elements of the protocol under test.

This rule relates to subclause 7.2.2, item c1 but is also be applicable to item b1 and b4. See also subclause 7.2.3.2, TC_Template, and example 13

Rule 2: Initialising test step in connection oriented context

For a connection-oriented protocol, one or more test steps should be provided to initialise the "data connection phase".

When the test purpose concentrates on the exchange of data PDUs, a stable situation is often needed where the IUT is not expected to send a data PDU spontaneously but waits for a data PDU from the tester. The objective of the test step mentioned in this rule is to reach such a stable situation.

EXAMPLE 12: Unsolicited transmission of data PDUs

In layer 2 of X.25, the "data connection phase" is called "information transfer phase". When this phase is reached, some implementations send a Restart Request packet, even as **called** party. The above mentioned test step would then be used to complete the restart procedure, when necessary, before entering the test body. The test step could later easily be adapted to other initialising operations, without affecting the test body.

This rule relates to subclause 7.2.2, items a3 and b2.

7.2.3.4 Rules for the test body

Rule 3: Use of test-steps

Test steps should be used each time a separate sub-function can be identified.

In accordance with principle 1 and 2, test steps should be used for each procedure that involves environment dependent events such as preamble, postamble. But, even in the test body itself, it could be very interesting to collect repeated behaviour lines in a test step. Often, such a repetition hides a sub-function of the protocol under test. The advantage of structuring the body using these functional test steps where possible, is obvious in the profile testing context.

This rule relates to subclause 7.2.2, items b3 and d.

Rule 4: Use of defaults

Use defaults in test cases and in test steps

This is in-line with the TTCN style guide, stating that only sequences of events directly related to the test purpose should occur in the test body.

Rule 5: Use of test components ("Concurrent TTCN")

When different PCOs exist, and sequences of independent events/PDUs (except for some synchronisation) can be isolated at the different PCOs, use "Concurrent TTCN" and parallel test components to separate the behaviour.

In order to separate behaviour lines properly, especially when re-use changing the PCO definition is expected, it is recommended to use "Concurrent TTCN". Then for each PCO, the behaviour is separated into a separate test component. A change in PCO definition then simply means replacing, removing, or adding one test component.

This rule relates to subclause 7.2.2, items b and c.

EXAMPLE 13: Upper tester treated as a test component

In this example the problem is to handle both versions of the test suite with upper tester or without upper tester. In this solution the possible upper tester is treated as a test component. Then, modifications affect only a single test step.

Initialisation of the test step is in both cases (Definition of Tpl_Test_Initialisation):

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1		CREATE(Upper_Tester,Tpl_UT_TestStep)			

In the first case, with an upper tester, the test step specifying the behaviour of the upper tester is (Definition of Tpl_UT_TestStep):

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1		UT? UT_event_1 UT! UT_event_2 CP ! UT_RESULT	event_1 event_2 UT_res(PASS)	 (P)	

In the second case, without an upper tester, the test step specifying the behaviour of the upper tester is:

Tpl_UT_TestStep: the UT behaviour is replaced by an implicit event

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1		<IUT! UT_Msg_2> CP ! UT_RESULT	 UT_res(PASS)	 (P)	

End of EXAMPLE 13

Rule 6: Qualify alternatives of allowed event lines

When a protocol allows different answers on a certain event, use test suite parameters associated to PICS or PIXIT items referring to the option, and put these as a qualifier in the event lines of the dynamic behaviour. Profiles restricting the allowed answers can then make use of the same test case by changing the value of the test suite parameter.

EXAMPLE 14: Typical alternative in X.25 layer 2

Test Case Name: TC_ST04_ERR
Group: X.25/DL4/INV
Purpose: Verify that in state 4 the IUT sends an SABM/P=1 or a DM/F=0 on error
Default: Def_ST
Comments:

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1	start	+Tpl_Preamble			
2	body	(produce an error)			
3		LT?SABM [SEND_SABM_ON_ERR]	SABM_P1	(P)	
4	end_1	verify state...			
5		LT?DM [SEND_DM_ON_ERR]	DM_F0	(P)	
6	end_2	verify state...			

For each implemented behaviour, the same test case can be used by giving the test suite parameters SEND_SABM_ON_ERR and SEND_DM_ON_ERR the corresponding value.

7.2.3.5 Rules for data order

Rule 7: Data PDU exchange in connection oriented protocols

One or more test steps should be defined to treat the exchange of data PDUs in the "data connection phase"

The number and contents of data PDUs that the tester has to send, and the number and contents of data PDUs sent by the IUT as a response to those, normally depends on the protocols implemented beyond the upper boundary of the IUT.

An IUT implementing SCCP e.g. in a GSM environment, could perform some local operations after receiving a data message from the tester, send a data message as response, then perform some remote operations, and send other data PDUs to the tester when these operations have been completed.

When a test step is used according to the rule, the step can be adapted to the protocol(s) involved, without changing the test body.

This rule is related to subclause 7.2.2, item b3.

7.2.3.6 Rules for data contents

Rule 8: User dependent data in constraints

If specific fields in constraints are subject to change because of a modification of the context in which the IUT is used, use a parameterisation technique to handle the change. Several techniques are available but the most efficient for re-usability is to use a TS parameter.

Whenever more than one protocol uses the same service provider (as normally is the case for protocols below OSI-layer 7), in the re-usability context the problem arises how to handle a PDU containing data coming from this embedding. This data has to be changed when a different embedding is used.

With TTCN this problem can be solved using structured constraints (the possibility of "hard-coding" the user data in constraints exists, but is really not recommended and contradicts the principle of re-use). This means a field of a constraint can be statically or dynamically assigned a value.

The TTCN style guide gives a broad discussion on this subject. In the style guide, the rules numbered 12c, 12d, 14a, 15a, 16b, 17b, and 18b are (partly) dealing with ASP and PDU constraint specification.

The first advice on this subject is to use rules 12c and 12d, which effectively say "use the ASP/PDU definitions of the service/protocol standard for defining ASP/PDU Types and Constraints in the test suite". This structured development will help the handling of further modifications in the PDU coming from a change in the embedding of the protocols (for example within a profile).

The other rules of the guide provide several techniques to parameterise the constraint.

a) Locally, in a test case, a fixed value can be assigned to the constraint field.

   This solution is not flexible: in order to change these constraint fields, every test case that uses this constraint has to be modified.

b) A test suite constant can be assigned to the constraint field in the test case.

   This solution is more flexible: Test Suite Constants can be redefined in one place. However, even in this solution, the constraint fields in individual test cases may have to be modified: one may want another Test Suite Constant to be assigned to a field.

c) A dynamic parameter can be assigned to the constraint field in the test case.

This solution, to make the constraints parameterisable, is the best and the most flexible. Test Suite Parameters and Test Suite Variables are dynamic parameters that can be used by any test case or test step in the suite. Dynamic parameters can be assigned to a constraint field directly in the behaviour line or indirectly as a test step parameter. Passing parameters to a test case is not allowed in TTCN. This problem however can be circumvented by defining a test case as a test step, and by then calling this test step in a one-line test case (this technique is already being used in the CTS-NM test suite and is supported in IS-TTCN). Example 15 elaborates on this.

This rule relates to subclause 7.2.2, item b4.

EXAMPLE 15: Usage of test step parameters to handle constraint field modification

In order to illustrate mechanism c) of the previous paragraph, a test case is converted to a test step, and is called from a newly created test case or from a test case table.

Step 1**:** Conversion of test case into a parameterisable test step

The following initial test case of SCCP using TCAP at the upper boundary will be converted.

NOTE: The constraints are parameterised.

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1		LT! UDT	UDT1(tcap_sent)		
2		LT? UDT	UDT2(tcap_recv)		

The test case is converted into a parameterised test step:

Test Step Name: TS_TC1 (user_data-sent,user_data-rcv: ASP_identifier)
Group:
Objective:

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1		LT! UDT	UDT1(user_data_sent)		
2		LT? UDT	UDT2(user_data_rcv)		

A newly defined test case then calls the test step with the same actual parameters that appeared in the first table of this example:

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1		+TS_TC1 (tcap_sent, tcap_recv)			

Step 2: usage of the parameterised test step

In this way, the usage of user data is lifted to the level of test step parameterisation, which makes modification very simple. Of course, the number of parameters necessary for a given test case can become quite large, but it should be noted that normally only a few of the constraints contain embedded data of upper layer protocols. If, however, many constraints are used, there is often a way of using the same embedded structure for several constraints.

Test cases constructed in this way can easily be grouped into a test case table. In the following example, a test case table containing three test cases is adapted for different embeddings. The following could be the version where SCCP upper boundary is accessible, then User_Data can be a simple characters string.

Test Case Dynamic Behaviours (Compact Table)			
Group: SCCP/routing tests **Default**: def_one			
Test case Name	**Purpose**	**Test step attachment**	**Comments**
TC1 TC2 TC3	route on GT route not on GT etc.	+TS_TC1(User_data,User_data) +TS_TC2(User_data,User_data) +TS_TC3(User_data)	

The same test cases can be adapted to a test configuration where the SCCP user reaction can only be managed via a TCAP layer. Then, it is necessary to handle a real transaction at the TCAP level. The problem here is to differentiate between the User Data and the transactions allowed for each user.

Test Case Dynamic Behaviours (Compact Table)			
Group: SCCP/routing tests **Default:** def_one			
Test case Name	**Purpose**	**Test step attachment**	**Comments**
TC1 TC2 TC3	route on GT route not on GT user unavailable	+TS_TC1(Begin_1,End_1) +TS_TC2(Begin_2,End_2) +TS_TC3(Begin_3)	(1) (2) (3)
Detailed Comments: (1) the service 1 operates on the user not accessible by Global Title (2) the service 2 operates on the user accessible by Global title (3) the service 3 is not accessible			

End of EXAMPLE 15

7.2.3.7 Rules for postambles

Rule 9: Use of postambles

Use a postamble at each "exit point" in the test body. The postamble should be a test step, and contain at least one exit test step.

This test step can later easily be modified to also bring the environment of the IUT into a stable state, if necessary, without affecting the test body

See also subclause 7.2.3.2, TC_Template.

Rule 10: Test step for assigning verdicts in "Concurrent TTCN"

When "Concurrent TTCN" is used, the test step used to handle the end of the test should contain the collection of the verdict from test components.

When a test case is designed in a multi-party context, then the exit test step of the postamble should collect the verdict of the several Test Components in order to assign the final verdict. If a re-use with a different Test Component organisation is intended, this will help the re-use of the test suite (one test step modification instead of each test case).

In a multi-party context this exit test step generally consists of the following actions:

a) Wait for verdicts of other components;
b) Do cleanup actions;
c) Assign a final verdict.

This rule is related to the situation of subclause 7.2.2, items b) and c)

EXAMPLE 16: A typical Exit test step when used in single party context

Definition of Tpl_Test_Exit

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1		[R= PASS] [R= FAIL] [R=INCONCLUSIVE]		P F I	

EXAMPLE 17: A typical Exit test step when used in multi-party context

Definition of Tpl_Test_Exit

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1		CP?UT_RESULT(RUT:=UT_RESULT) [RUT = "PASS" AND R="PASS"] [RUT = "FAIL" OR R="FAIL"] [RUT ="INCONCLUSIVE" AND R ="INCONCLUSIVE"]		 P F I	

7.3 Rules for modularization of test suites

7.3.1 Basic concepts of Modular TTCN

A TTCN module can be seen as a collection of TTCN objects, available to be imported and used by other Modules or by test suites as illustrated in figure 7:

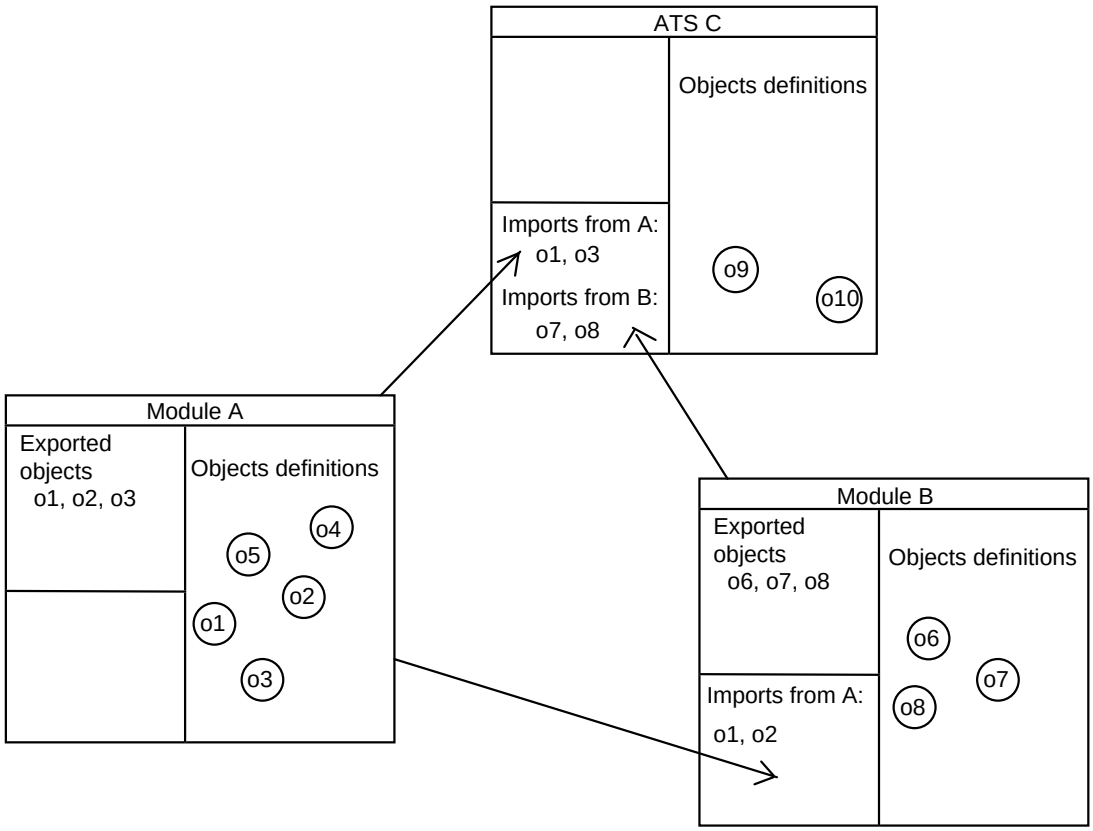


Figure 7: Import/Export in modular TTCN

The structure of a module is roughly the same as the structure of a suite: the difference between a module and a test suite is that a module embeds a module overview containing the list of exported objects. For still open issues with Modules see annex C.

7.3.2 Basic concepts of modularization

7.3.2.1 General

There are a lot of possibilities for splitting an ATS into Modules, or for defining Modules. For example, the declaration part or dynamic part can be placed in a Module, or, from a different point of view, test cases for testing a specific functional unit of the protocol under test can be placed in a Module.

A number of questions concerning definition of Modules can be derived from figure 8 below:

a) What should be the form of Modules? Are there requirements or criteria for the shape of module, or on the type of TTCN objects found in Modules?
b) Which TTCN objects are suitable to be placed in Modules, which not?
c) "How far" should a test suite be split up?
d) Should a "standard" way of splitting up a test suite in Modules be proposed? (Always split up in certain well-defined parts) If not, what are then possible criteria for splitting a test suite?
e) A list of module types, as complete as possible?

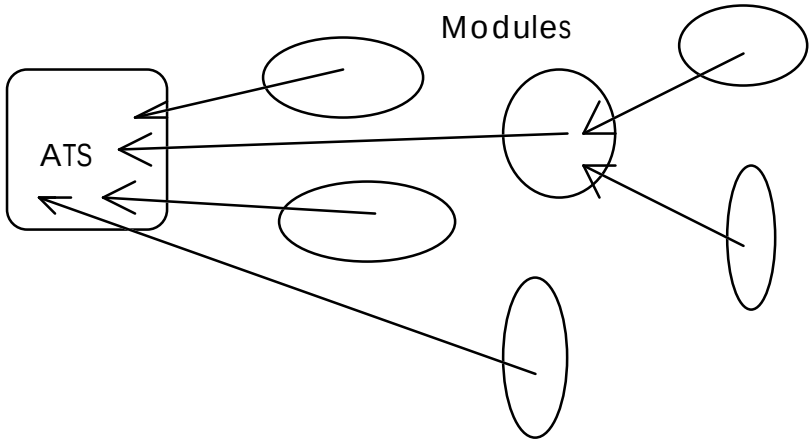


Figure 8: Modularization of an ATS

The above questions are addressed by developing rules, principles, and criteria, which are inspired by software development principles. They aim to make Modules re-usable and maintainable. Reusability and maintainability should be kept in mind when defining Modules.

7.3.2.2 General rules for the definition of Modules

If it is not possible to express in a single simple rule stating which kind of objects belong to the module, there is a large chance that this module will not be re-usable by someone else.

Rule 11: Classifying rule for module definition

There should be one clear rule for the definition of a module.

In order to facilitate maintenance, TTCN objects expecting change and TTCN objects expecting no change should be placed inside different Modules.

Rule 12: Separate TTCN objects to be changed

In order to facilitate maintenance, separate TTCN objects expecting changes and TTCN objects expecting not many changes in different Modules.

The dependencies between Modules, introduced by the import, should be clearly identified and minimised. A diagram showing the dependencies could be a good support for that. Cycles in this diagram, even if they are allowed by Modular TTCN, should be avoided. Figure 9 below gives examples of such diagrams.

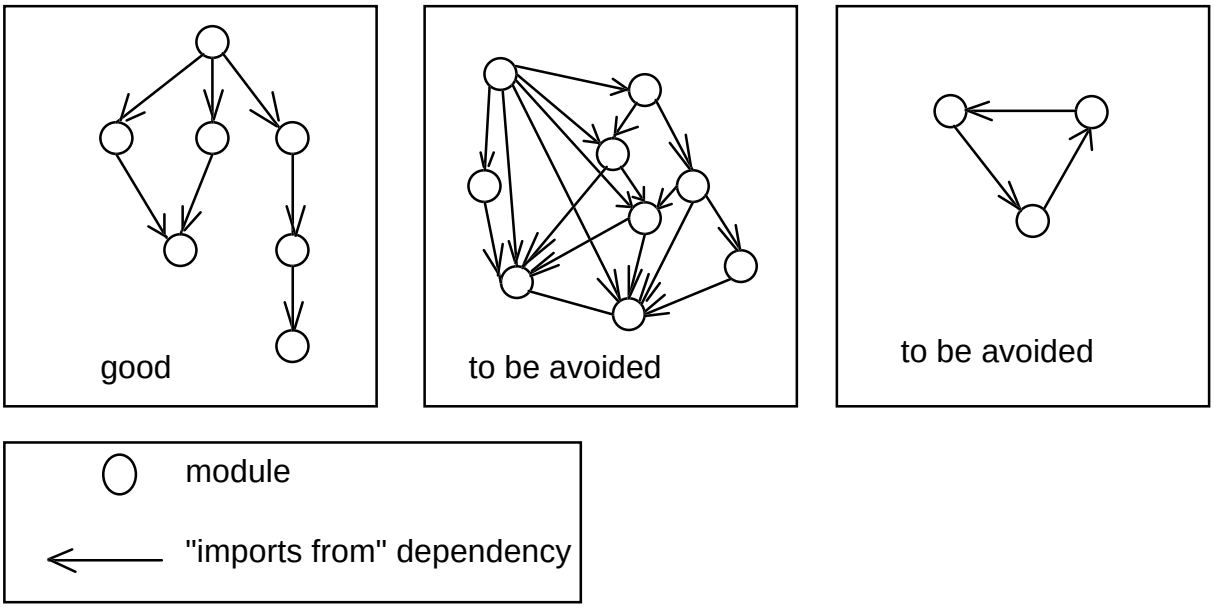


Figure 9: Examples of module dependency diagrams

Rule 13: Dependencies between Modules

Identify and minimise the dependencies between Modules. Avoid dependency cycles.

In a modular organisation the relative size of Modules is not relevant. However, a large module often hides a lack of modularisation. Small Modules have the advantage to be easier to manipulate and to understand, and easier to treat by tools. Small Modules could lead to an explosion of the organisation tree but the decision to include it in another module should be taken regarding the coherence of the organisation.

Rule 14: The module size should be reasonable

The size of a module should remain reasonable. Modules containing less than 10 tables or more than 1 000 should be avoided.

7.3.2.3 Modules interfaces

For the purposes of this subclause, the module interface is defined to be composed of:

a) The **upper interface,** constituted by the list of exported objects (including the list of formal parameters, when applicable).

   The upper interface is related to the questions: "Which objects can be used by others? How are they used?".

b) The **lower interface** is constituted by the list of external objects. The lower interface answers the question: "What are the objects avoided to be defined in this module?".

   NOTE: Point b) is only applicable if external objects are allowed for a module (see the different views expressed in subclause 6.5; see also annex C). Rule 16 on lower interface of Modules relies on this assumption.

The interface of a module should be clearly described in order to facilitate its usage by other ATSs or Modules. If an exported object takes parameters, e.g. parameterised constraint, what is expected for each parameter should be written in the Comments column.

Rule 15: Document the upper interface

All the objects to be available for use outside the module should be thoroughly described in the Module Overview. For each parameterised object a comment should state the usage of each parameter.

To make the lower interface clear, the external objects together with their parameters (if applicable) and the way in which they are used should be described in the Module Overview. For each reference to this object in the module, a comment should clearly state that this object is external.

Rule 16: Document the lower interface

All the external objects with their required parameters should be listed in the detailed comments of the Module overview. Each reference to an external object in the module should be commented.

The following example illustrates rules 15 and 16.

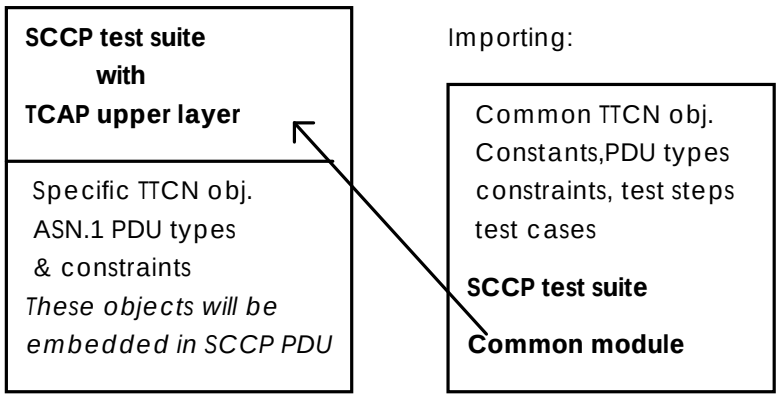
EXAMPLE 18: Documentation of Modules interfaces

The previous SCCP/TCAP example (example 15) is treated here in a different way, inside a common/specific part combination.

Remember that TCAP structures are embedded in the SCCP PDU (in SCCP User Data part). The difficulty for the definition of a test suite for SCCP comes from the fact that it is not possible to be sure that an access will be available at the upper layer boundary. Often, but not always, the TCAP layer is completely linked with the SCCP layer. The remote testing method is used there, but it is necessary to define a clear interface between the two layers at the TTCN level in order to be able to manage any kind of upper layer or services above SCCP.

In this example a SCCP test module that will be the "common part" is defined. Then this module is imported in a specific part that is also the resulting ATS "SCCP test suite using a TCAP remote upper service" (see also in subclause 7.3.3.3 describing how to organise Modules).

The organisation is the following:



First define (elements of) the SCCP common module:

Test case name: TC11221
Group: Route on GT
Purpose:

Nr	Label	Behaviour description	Constraints ref.	Verdict	Comments
1	start	.			
2		.			
3	body	LT! UDT	RtOnGT_Type1		(1)
4		LT?UDT	Up_Layer_MsgAck	(P)	(2)
		.			

Detailed Comments: (1) (2) this is an SCCP expression of the test completely independent from any upper layer. The remote test method is assumed.

PDU Type Definition

PDU Name: UDT
PCO Type: MTPSAP
Comments:

Field Name	Field Type	Comments
RoutingLabel	Rt1	
MessageType	BITSTRING[8]	
ProtocolClass	BITSTRING[8]	
Pt1	OCTETSTRING[1]	
Pt2	OCTETSTRING[1]	
Pt3	OCTETSTRING[1]	
Called Address	Cdpa	
CallingAddress	Cgpa	
SccpLength	OCTETSTRING[1]	(1)
SccpUserData	SccpUserData_Type	(2) this TYPE is EXTERNAL

Detailed Comments:
(1) This field contains the length of the SCCP User Data part
(2) This field uses a type that is not defined in this module. It must be defined in a test suite importing the current module.

PDU Constraint Declaration

Constraint Name: RtOnGT_Type1
PDU Type: UDT
Comments: this constraint has an external user part

Field Name	Field Value	Comments
.	.	
.	.	
Called Address	IUT_GT_Type_1_Address	
CallingAddress	TesterAddress	
SccpLength	CALCLENGTHOF(ULD_SentByTester)	(1)
SccpUserData	ULD_SentByTester	(2)

Detailed Comments:
(1) a Test Suite Operation is invoked because it is not possible to known in advance the length of the user data part.
(2)This field is filled with a constraint imported from another module. ULD stands for Upper Layer Data.

PDU Constraint Declaration		
Constraint Name: Up_Layer_MsgAck **PDU Type**: UDT **Comments**: This constraint has an external user part		
Field Name	**Field Value**	**Comments**
. .	. .	
Called Address	IUT_GT_Type_1_Address	
CallingAddress	TesterAddress	
SccpLength	CALCLENGTHOF(ULD_for_MsgAck)	(1)
SccpUserData	ULD_for_MsgAck	(2)
Detailed Comments: (1) A Test Suite Operation is invoked because it is not possible to know in advance the length of the user data part. (2)This field is filled with a constraint imported from another module. ULD stands for Upper Layer Data.		

The SCCP common module will then define an export table like the following. This table is the expression of the **Upper Interface**:

Module Export			
Module Name: SCCP_Common_Module **Comments:** This module should be totally included in an ATS in order to obtain a complete SCCP test suite.			
Object Name	**Object Type**	**Page Nr**	**Comments**
.	.	.	
TC11211	TestCase	41	
TC11221	TestCase	42	
	.	.	
Detailed Comments: Definition of the module upper interface: This module contains the definition of test cases for all test purposes applicable to the SCCP protocol, as they are defined in TP_SCCP... The module also contains all other definitions or declarations necessary to express the test cases, except for some constraints of the user data field of SCCP PDUs (and, associated to this: the length of that field). These constraints are listed in the External Objects table of the module.			

The "SCCP common module" contains an External Objects table like the following:

External Objects		
Object Identifier & Parameters	**Object Category**	**Comments**
SccpUserData_Type		(1)
ULD_SentByTester		(2)
ULD_for_MsgAck		(2)
Detailed Comments: Definition of the module lower interface: (1) The type associated to the SCCP user data field is left open to allow any protocol above SCCP and any type within this protocol. (2) The constraint value is also left open.		

Then the test suite "SCCP_Tests_with TCAP _UpperLayer" is defined.

Import			
Module or Suite Name:	SCCP_Common_Module		
Reference:	Annex X of ETS 300 xyz (1995), Version 2.1		
Comments:	The whole SCCP_Common_Module is imported with no name change		
Unit Category	**Unit Reference**	**New Object Identifier**	**Comments**
TTCN_Module			
Detailed Comments: The imported module contains all SCCP test cases, except for some upper layer dependent constraints for SCCP user data, and their types.			

ASN.1 PDU Type Definition	
PDU Name:	SccpUserData_Type
PCO Type:	SCCPSAP
Comments:	TCAP BEGIN PDU

Type Definition

```
[APPLICATION 2] IMPLICIT
SEQUENCE {
        otid            OrigTransID,
        dialog_portion  Dialogue_portion OPTIONAL,
        components      ComponentPortion OPTIONAL
}
```

Detailed Comments:
BEGIN message type declaration. The tag value and the length of transaction portion result from the usage of APPLICATION and IMPLICIT SEQUENCE.

ASN.1 PDU Constraint Declaration	
Constraint Name:	Beg_Invk_TestOp
PDU Type:	SccpUserData_Type (TCAP BEGIN PDU)
Comments:	

Type Definition

```
{otid           OTID_1,
 components     {
                  invoke  {   invoke_ID       INVKID_1,
                              operation_code  {local_value TEST_OP_1}
                          }
                }
}
```

Detailed Comments:
OTID_1, INVKID_1, TEST_OP_1 are Test Suite Constants.

The definitions table for of the PDU End_RRL_Ack not expanded here. They are based on the same principles.

7.3.3 Criteria and architecture for modularization

7.3.3.1 General

The introduction of modularity in TTCN has an impact on ATS development methodology. The modularisation of ATSs provides advantages in the two following directions: allow a fragmentation approach of the TTCN developments; and share these developments within the testing community. In order to reach these two aims, the definition of Modules should be the result of a methodical approach of the testing problems.

Needs for fragmentation stated by TTCN ATS writers should not be the only aspect that influences the arrangement of Modules. ATS designers should be aware that many other criteria can be found. The following subclauses list some of these existing criteria and propose to classify them by levels.

A level is a class of criteria that leads to the definition of a module classification. The resulting Modules could be themselves modularised following another level but this approach is left to the judgement of ATS designers.

Three levels will be developed here:

1) The functional or service level;
2) The organisational level;
3) The language level.

7.3.3.2 Functional level

The functional level induces a partition of test suites into several functional parts. For a given protocol, these parts may be formed by services provided by the protocol under test. In the actual TTCN ATS, this modularisation is performed through the tree organisation of the test cases (groups).

This classification can be performed at the test suite structure level when working on the test purposes list. This kind of classification could be performed by the evaluation of PICS/PIXIT of the protocol under test.

The following list gives some classification criteria that could be used for the definition of Modules. The application of one of these criteria, which must be chosen depending on the situation, will result in a set of TTCN functional Modules. A functional module can be a complete test suite with test cases adapted to a functional classification, or an incomplete TTCN module (e.g. without test cases).

CRITERION 1: Services the protocol under test provides

For example, ATSs for SCCP can be partitioned in: SCCP control services, SCCP transfer services, SCCP routing services, SCCP replicated subsystem services, etc.

CRITERION 2: Services provided by the lower layers

For example, different kinds of interfaces on the same protocol. The ISDN layer 3 test suites for Basic Access and Primary Rate Access use a common set of test cases.

CRITERION 3: Categories of protocol users

For example, in SS#7, the testing of TCAP is influenced by the category of equipment containing the IUT (IN-SSP, IN-SCP, GSM-HLR, GSM-MSC, etc.). A set of test steps for each category of equipment could be defined and gathered in a module in order to easily produce several test suites.

CRITERION 4: Protocol states

For example, in ISDN layer 2 test suite the organisation clearly follows the protocol states reached in the preamble.

CRITERION 5: Direction in non-symmetric protocols

For example, the ISDN layer 3 test suites for Network side and User side use a common set of test cases. Here the functions provided by one side and the other should be clearly identified.

CRITERION 6: Standardised Modules for ASP and PDU type declarations

For example, it can be assumed that standard declarations of PDU and ASP types could exist for each protocol. This would introduce the concept of test oriented materials in standards (i.e. TTCN module following a protocol standard).

CRITERION 7: Test step libraries

For example, the software development domain gives good examples of what could be provided to users in the form of a functional library. In TTCN, test steps could be considered functions. It could be assumed that some standard library of test steps could exist (e.g. ISDN layer 2 standards state preambles).

7.3.3.3 Organisational level

The organisational level is a class of criteria related to an ATS writer's point of view. These kinds of criteria can be applied to define the best partitioning of Modules in order to minimise the development effort.

A very simple set of criteria for the organisation level is introduced here. This organisation allows a design where Modules are shared between several ATSs (or Modules).

Each ATS (or module) can be divided into Modules (sub-Modules) which are here called a "part". Three categories of parts are defined in figure 10 : global; common; and specific.

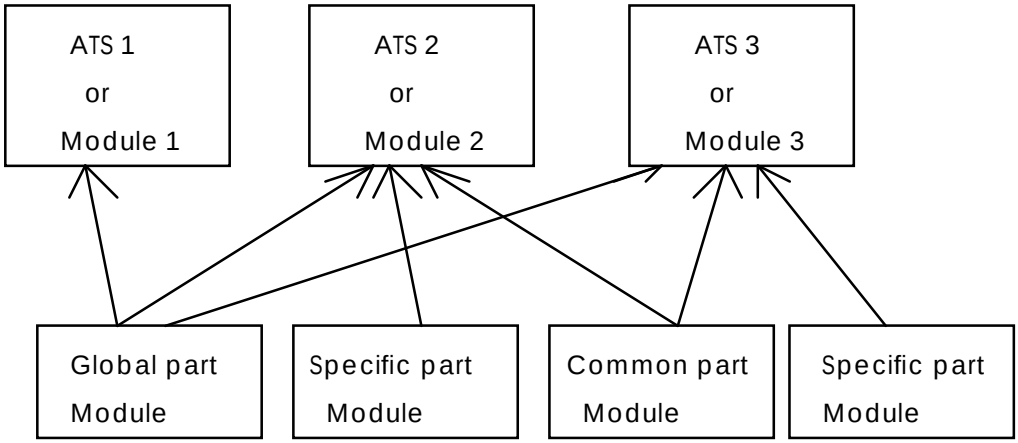


Figure 10: An organisation sample

A global part module concerns TTCN entities that could be shared by all other ATSs (or Modules) in a project. Such a part will typically contain the definition of some global constants but could also contain test suite parameters, PDU type declarations and, why not, some global test steps.

A common part module is a set of TTCN objects shared between several ATSs (or Modules) but not all of them. This kind of module will typically contain test cases when used in conjunction with the criteria 6 and 7 of subclause 7.3.3.2, but this implementation of common part by module could also support the common/specific paradigm.

A specific part module contains all TTCN objects that could not be found in another module. In many cases the specific part module will be the ATS itself where other Modules (i.e. global and common parts) will be imported. This produces the simplified organisation sample, as shown in figure 11:

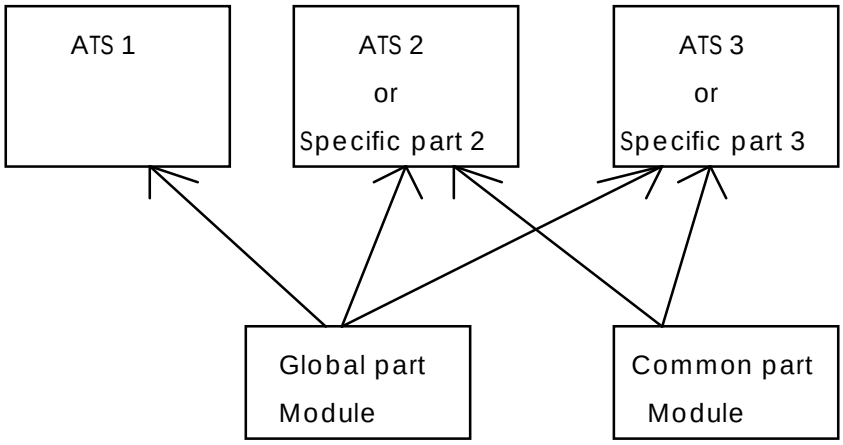


Figure 11: Simplified organisation sample

7.3.3.4 Language level

The language level is a naive class of criteria that is based on a classification of TTCN objects by the TTCN part or sub-parts they are belonging to, such as:

- constant declarations;
- test suite variables declarations;
- PCO declarations;
- timer declarations;
- ASP type definitions;
- PDU type definitions;
- alias definitions;
- constraints;
- test cases;
- test steps;
- defaults.

A distribution of TTCN objects into Modules without any criteria from other levels is useless and will lead to a standard TTCN ATS (i.e. not modular). It can be assumed however that some types of objects like, for example, ASP and PDU type declarations for a given protocol, could be a standardised object that could be separated from the others and widely re-used.

The language level can be used for the distribution of objects by categories into Modules. For example, a global module contains no test case at all while a module used to share several common test cases between two Modules contains only a set of test cases.

Many kinds of Modules can be found by combining several TTCN categories. The following list gives some examples of combinations that can be produced:

1) a general usage "TTCN library" including global constants, variables, test steps;

2) a test case library including the test steps that are referenced;

3) a protocol standards TTCN module including constants declarations, constant values, variables declarations, variables initialisations, ASP and PDU types definitions and some useful parameterised constraints.

Many other arguments for splitting an ATS into Modules by category could be derived from the rules of subclause 7.2 (rules for re-usability of test suites). The following tables give an idea of possible implementation of the rules in the modular scheme:

Table 2: A modular approach to the rules of subclause 7.2.

Principle 1	Place preambles, postambles, other test steps and test components in Modules
Style guide Rule 12c	Place ASP and PDU definitions from service/protocol standards in a module.
Style guide Rule 12d	Split the ATS in several Modules corresponding to the structure found in the service/protocol standard.
Style guide Rule 15a	Put base constraints in a module. Use an existing base constraint module.
Style guide Rule 16b	One single module with parameterised constraints can be used for different profile environments.
Style guide Rule 18b	Place all test suite constants in a single module (like in a C program, where usually an include file exists which contains all constants of the program).
Rule 1	Create a module with preambles. Inside these preambles, initialising test steps will exist and could be collected in a different module. Check if the initialising test steps fit with a predefined library of test steps Check if the preambles match with a pre-defined library of preambles
Rule 4	Create a module with default trees Check if a predefined default trees library exist
Rule 5	Organise Modules by PCO if you think your ATS is likely to be used in another configuration. In Concurrent TTCN a responder can be implemented in an external module.
Rule 6	Create Modules with test suite constants that get their values from ICS or IXIT.
Rule 7	Exchanges during the data connection phase can be placed in a single module.
Rule 8	Use Modules to handle the possible modification of embedded data
Rule 9	Define a module containing test steps with exit points and errors handling.
Rule 10	Use Modules to handle some generic test steps in Concurrent TTCN.

7.3.3.5 Final architecture including the three levels

Finally a possible complete organisation of Modules follows in figure 12:

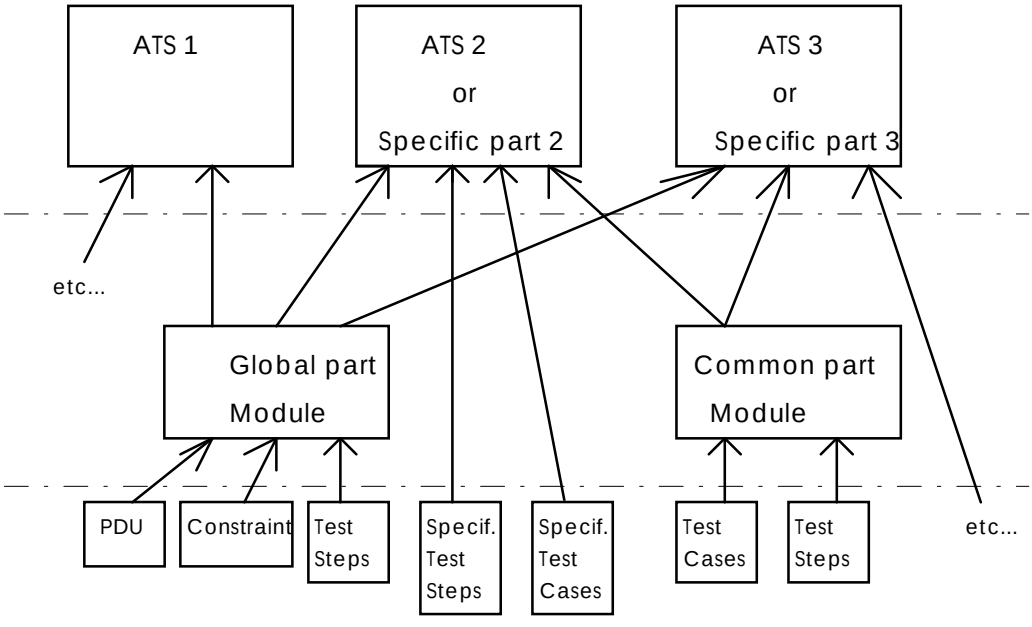


Figure 12: The three level organisation sample

It should be noted that such a module architecture can lead to defining TTCN ATSs that are almost only consisting of import tables.

A practical solution is to consider the third level (language level) only as a designer level which is already included in the organisation of a module. Then the example organisation looks like figure 13:

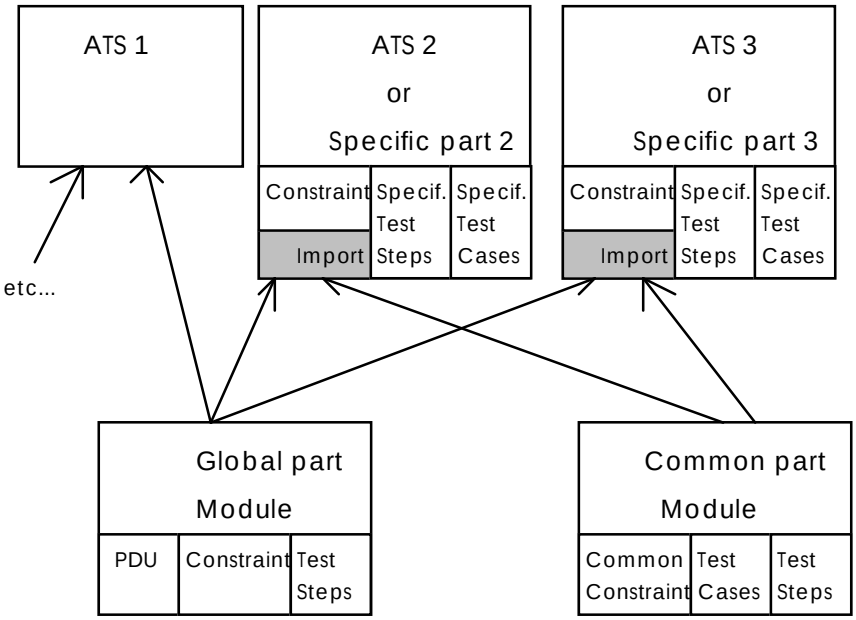


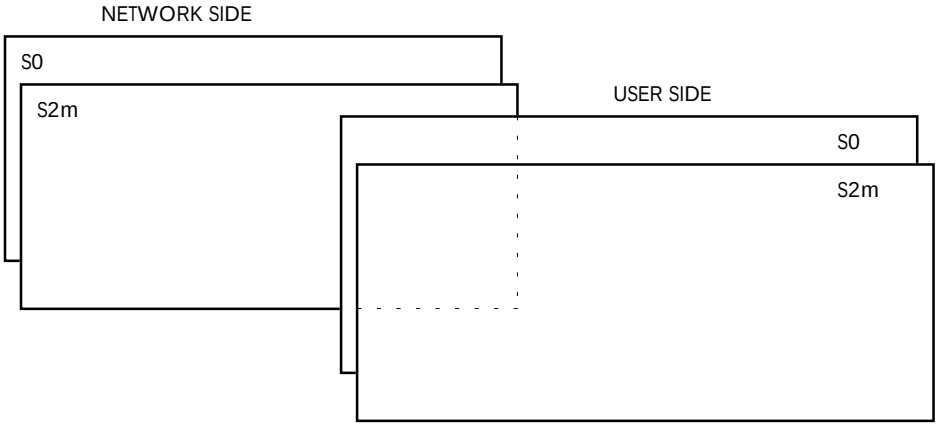
Figure 13: The final TTCN organisation

The following example is taken from real ATSs:

EXAMPLE 19: National ISDN ATSs

For the Basic Call in the French public ISDN called (RNIS), 4 test suites have been defined corresponding to the combinations of user side/network sides and S_0/S_{2m} interfaces. This work is already done, but imagine the organisation that could have been taken using modular TTCN.

These TTCN test suites overlap as follows:



A common part between all test suites can be defined. Two main specific parts appear with network side and user side that could also be divided into a common and two specific parts for S_0/S_{2m}.

Concurrently, these basic call TTCN test suites define many objects that are widely re-usable by other test suites such as Supplementary Service test suites. Then, it seems very interesting that a global ISDN library of TTCN object could arise from the following organisation.

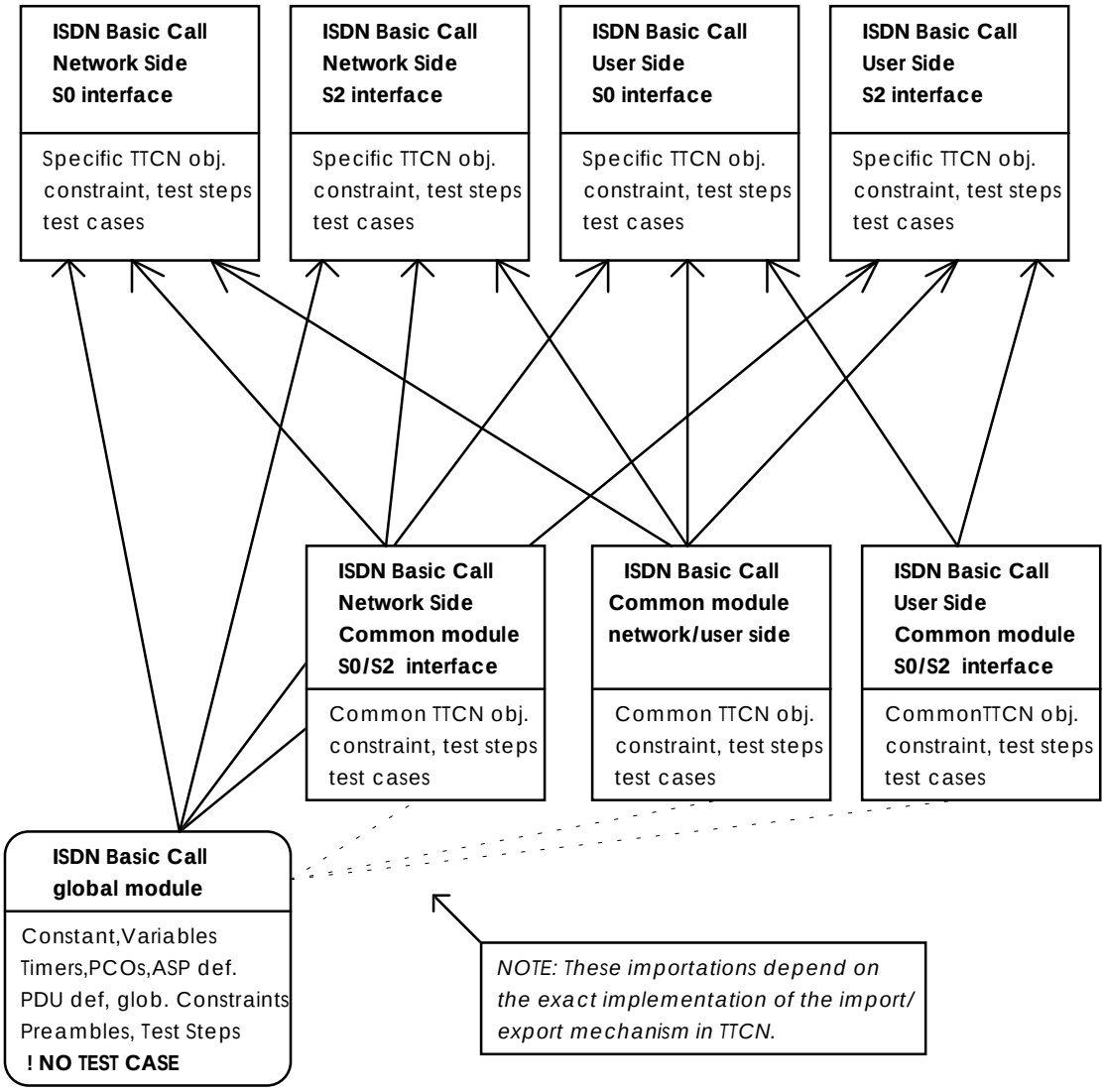


Figure 14: The final module organisation

7.4 Rules for the combination of Modules into test suites

When a new ATS has to be defined with the use of TTCN Modules, a number of questions are important:

a) which Modules should be used?
b) where can these Modules be obtained? This question is related to module management issues (see clause 8).
c) how to combine these Modules into a Modular TTCN ATS?

These questions are addressed in following subclauses.

7.4.1 Choice of Modules

A number of different ATS development situations exist. These are briefly summarised here, and are addressed in following subclauses:

a) an ATS has to be developed "from scratch";
b) an ATS has to be developed by adapting an existing Modular ATS;
c) an ATS has to be developed by adapting an existing (non-modular) ATS;
d) Modules have to be developed.

For b) and c), also different adaptation levels exist: an ATS may be adapted on each of the possible adaptation points mentioned in subclause 7.1:

1) adaptation of the protocol itself: this may mean different test cases, modification or rewrite of test cases;

2) adaptation of the upper layer usage: this may mean different user data, different preambles;

3) adaptation of the lower layer usage: this may mean different preambles, different data exchange test steps;

4) adaptation of the protocol role: e.g. acting as a responder instead of as an initiator which has implications on connection set-up test steps.

7.4.1.1 Development "from scratch"

When a totally new ATS has to be developed, the only Modules that could exist are very "general" ones containing for example PDU and ASP definitions. The remaining test material has normally to be developed directly.

Rule 17: ATS development from scratch

When an ATS has to be developed from scratch, check whether usable Modules exist, particularly PDU and ASP definitions. While developing the remaining testing material explicitly, use the methods of subclause 7.3 to create new Modules.

7.4.1.2 Development by adapting an existing Modular ATS

It is supposed that the existing Modular ATS is written according to the rules in subclause 7.2. Depending on the type of adaptation, Modules may be re-usable in different ways:

a) Not all TTCN objects of each module are imported.

b) Some extra TTCN objects should be defined on some Modules. This means a module grows, or a new module of the same type has to be created as an "additional module".

c) Additional or replacement Modules may have to be created (embedding dependent information, constraints, user data constants). This can be easy if the rules in subclause 7.2 have been followed.

Basically, rule 17 also applies to this situation, except that less testing material has to be developed explicitly.

7.4.1.3 Development by adapting an existing (non-modular) ATS

There are several possibilities. The existing ATS may be written according to the rules in subclause 7.2, so it may be more, or less, re-usable, or may be written in a very specific way for one embedding (non-re-usable).

In the first case, the procedures of subclause 7.3 can be followed to "distribute" the material of the test suite into Modules.

In the second case, the rules of subclause 7.5 should be applied for rewriting or reusing an existing ATS.

7.4.1.4 Development of Modules

Development of separate Modules may become an important activity in the future. It is thinkable that very "general" Modules will be developed, e.g. containing the maximal set of PDUs for a given protocol.

7.4.2 Combination of Modules into a Modular TTCN ATS

Modules that have been identified as described in subclause 7.4.1, have to be merged into a Modular TTCN ATS. This means defining links between Modules and ATS through the Module Overview and import tables of Modular TTCN.

For this situation, the "converse" rules (rules 15 and 16) of the "Module Interface" subclause (7.3.2.3) apply, which is summarized in the following rule 18:

Rule 18: Document the import tables

Use the comments entries to describe where Modules have been imported from, what has been imported, and how the importation has been performed. Do the same for imported objects.

7.5 Re-use of existing ATSs

The previous subclause shows how ATSs or Modules can be built in a modular way, in order to be able to re-use them. This subclause explains how to proceed when someone wants to re-use an ATS that was not written in a modular way. This is especially the case when it has not been written in Modular TTCN. Two cases are distinguished:

a) The ATS does not need to be rewritten. This is, for example, the case if:

   1) Import from test suites is allowed (see annex C);
   2) The test suite has some characteristics
      that make it suitable for re-use (see subclause 7.2).

b) The ATS needs to be rewritten.

7.5.1 Re-use of existing ATSs without rewriting them

If the assumption is made that import from a test suite is allowed (and the suite implicitly exports all it defines), it is possible to re-use ATSs written in IS-TTCN (not in Modular TTCN), by importing objects from this suite into a new modular suite.

This may however be difficult because such an existing test suite has not been designed to be re-used.

The first step in re-using material of the existing suite is to identify what can be re-used. To be of some interest, the parts that can be imported should have a significant volume. Otherwise, other possibilities should be envisaged (see subclause 7.5.2).

Rule 19: Import without rewriting only in case of significant re-use

Use the technique of importing from a ATS only if a significant amount of the objects can be re-used.

7.5.2 Re-use of existing ATSs by rewriting them

7.5.2.1 Context of applicability

The question, who is responsible for rewriting existing ATSs into a re-usable form, is addressed in clause 8.

The purpose of this rule is to redefine an existing test suite in a modular way, by cutting/pasting and rewriting elements from the existing test suite and putting them into Modules as illustrated in figure 15:

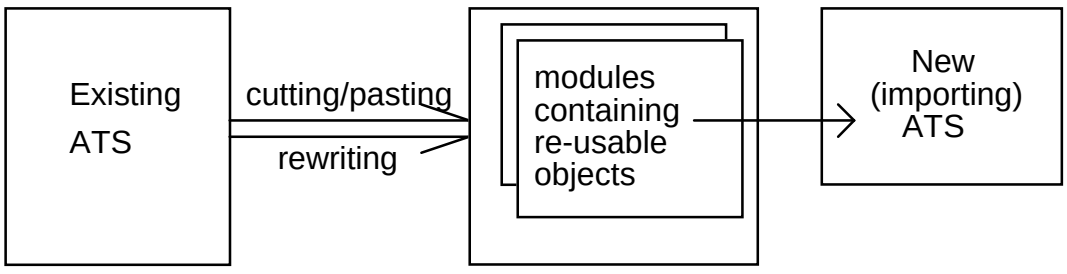


Figure 15: Re-written modular ATS

A test suite can be made suitable for re-use by splitting it into Modules according to the rules expressed in subclause 7.2. This step should be done in such a way that the modularised ATS is functionally equivalent to the existing one. As this operation performs significant changes on the ATS, the ATS should then be validated.

Rule 20: Re-validate modularised ATSs

An ATS rewritten in a modular way should be functionally equal to the original test suite.

Two ways of modularization basically exist:

a) Directly placing elements of the existing test suite in Modules (without modification). E.g. the test step section, the constraints section.
b) Redefining elements of the ATS (according the rules in subclause 7.2), and then creating Modules for them. E.g. parameterise a constraint, then put it in a module.

The first approach is discussed in subclause 7.5.2.2, the second in subclause 7.5.2.3.

7.5.2.2 Rules for directly placing elements of the existing test suite in Modules

When certain elements of the existing test suite are in an appropriate form (according the rules in subclause 7.2), they should be placed in Modules in modified form. Examples of such possible modifications are:

a) a test case in which preamble and postamble have been changed to parameterised test steps;
b) a constraint changed into parameterised form.

7.5.2.3 Rules for redefining elements of an existing test suite

If elements of an existing test suite have to be redefined, this redefinition should be performed following the rules defined in subclause 7.2.

7.6 Re-use of some existing test suite parts using the modular re-use style

As already described in clause 5, a number of re-use techniques exist. The intention of this subclause is to indicate how test suites written using these re-use techniques can be used in the Modular re-use style.

7.6.1 ROSE-technique, ACSE-technique, presentation-technique

Test suites and parts developed using these techniques are likely to be convertible to TTCN Modules. Probably the addition of import and export tables will be sufficient.

7.6.2 Session-technique

It will be difficult to capture the intuitive aspects of this technique in TTCN Modules: a conversion may be possible, however, an interpretation of these intuitive aspects is necessary.

7.6.3 CIST-technique

The CIST technique extends the ROSE technique. Those features of the CIST technique which are related to modifications, like the RAS-symbol, cannot be expressed in Modular TTCN.

7.6.4 Conclusion

Both the CIST technique and the Session technique require pre-processing to convert parts written in these techniques to Modular TTCN. The ROSE/ACSE/Presentation techniques are possibly easier to convert to Modular TTCN.

7.7 Profile testing and modularity

The profile testing methodology has been defined in ISO/IEC 9646-6 [8], and, for use according to the European standardisation, in ETS 300 406 [14]. According to ETS 300 406 a profile test specification (PTS) is made up of three parts:

1) the profile test specification summary (PTS-S);
2) the profile specific test specification (PSTS); and
3) the system conformance test report proforma (SCTR) tailored to the profile.

This subclause is mainly related to 1) and 2). Among other items a PTS consists of references to other standards, particularly existing ATSs, and it may contain profile-specific testing material in the PSTS which has to be combined in some way with the existing ATSs. The test cases defined in the PSTS can be divided in two classes:

a) additional test cases related to a particular protocol of the profile;
b) profile specific test cases not being related to a particular protocol of the profile (but to the "interworking" of two or more protocols).

The test cases in a) and b) often need some additional definitions like constraints or selection expressions. The combination of the additional test cases and the new definitions with the existing ATSs referred to in the PTS has been performed in an informal way. Modularity could help to put this in a more formal way.

To include the testing material related to a), profile-specific ATSs could be defined, which mainly import an existing ATS for a given protocol covered by the profile, and define only the profile-specific additional testing material explicitly. The test cases not applicable to the profile could be deleted in the Import table.

In situation b), some testing material necessary to define the profile specific test cases could be imported from ATSs referred to in the PTS, the remaining testing material including the profile specific test cases themselves being defined explicitly.

For both cases a) and b), the resulting profile specific TTCN test suites could be included in annexes of the PSTS, like a TTCN ATS for a given protocol is included in an annex of an Abstract Test Specification according to ETS 300 406 [14].

NOTE: The procedures proposed above are only possible if import from test suites is allowed in Modular TTCN (see point b) of annex C).

8 Consequences of modular TTCN on test specification methodology and standardization organizations

8.1 General

Several assumptions have been made, respectively, alternatives have been stated in this ETR, the realization of which is presently unclear. The most important of them are repeated here in form of questions. The consequences indicated in the subsequent subclauses will depend on which answers will be given to these questions in the future:

Q1 Will the concept of modular TTCN finally be introduced in ISO/IEC 9646-3?
Q2 If yes: what will the concept of module exactly be? Two possible but extreme positions have been indicated in subclause 6.5, as View 1 and View 2. Other positions are also possible. See also annex C.

8.2 Consequences on ISO/IEC 9646

8.2.1 Consequences on ISO/IEC 9646-2

Draft amendment to ISO/IEC 9646-2 [2] introduces the concept of common and specific parts, which should be aligned to the concept of Modular TTCN if Q1 is answered with "yes".

8.2.2 Consequences on ISO/IEC 9646-3

Modular TTCN is expected to be included in ISO/IEC 9646-3 [3], depending on the answers to questions Q1 and Q2, above.

The envisaged extension extends the 1992 edition of ISO/IEC 9646-3 [3], added with the amendments 1 [4] and 2 [5].

8.2.3 Consequences on ISO/IEC 9646-6

Depending on question Q1, and particularly question Q2, Modular TTCN could have an influence on profile testing methodology, by expressing profile-specific testing material in TTCN Modules and including these Modules in profile test specifications. See also subclause 7.7.

8.3 Consequences on the European test specification methodology

a) ETR 141 [11]: "The TTCN style guide" - is expected to incorporate or refer to (part of) the rules defined in this ETR.

b) ETS 300 406 [14]: "Protocol and profile conformance testing specification: Standardisation methodology" - the usage of Modular TTCN is expected to be included.

8.4 Managing testing material

A short introduction of how testing material may be available has been given in subclause 6.2 (see figure 4). ATSs, TTCN Modules and Profile Test Specifications will have to be managed. For the purposes of this ETR, the management in standardization organizations, particularly in ETSI is the main subject.

There should be an answer to questions like:

1) who takes care of rewriting or converting existing ATSs so they will become modularised?

2) when new ATSs or Modules are written: who is responsible to decide what Modules may be imported and what may be exported?

3) how can Modules be uniquely identified for import (locally, maybe also internationally)?

4) how can a user find the Modules serving his needs?

5) who takes care of maintaining Modules (versions)?

NOTE: An ASN.1 library definition and an ASN.1 library index are currently being proposed in ETS 300 655 [12] and ETR 210 [13], respectively. These documents could serve as a base for similarly building dedicated test specification libraries.

8.5 Consequences on TTCN Tools

Quite different kinds of TTCN tools exist, e.g.:

a) Editing tools, exporting TTCN.MP code;
b) Compilers;
c) Test tools, containing compilers (or not).

If Q1 is answered with "yes", it can be expected that editing tool manufacturers will follow and implement Modules and importing features. The way of implementation will of course depend on how Q2 will be answered.

How ever Modular TTCN will be defined, editing tools of kind a) are expected to be able to convert test suites containing import tables into TTCN.MP code without import tables, i.e. complying with the existing IS-TTCN specification.

Manufacturers of tools of type b) or c) will therefore not necessarily follow the development of Modular TTCN.

Annex A: List of rules

To enable an easy and quick cross-reference to all rules being defined in the body of the ETR, table A.1 below provides the complete list of the rules including rule-number and the number of the page on which it appears:

Table A.1: List of rules

N^{O}	Rule	Page
1	Test step in preamble	27
2	Initialising test step in connection oriented context	28
3	Use of test-steps	28
4	Use of Defaults	28
5	Use of test components ("Concurrent TTCN")	28
6	Qualify alternatives of allowed event lines	29
7	Data PDU exchange in connection oriented protocols	30
8	User dependent data in constraints	30
9	Use of postambles	32
10	Test step for assigning verdicts in "Concurrent TTCN"	32
11	Classifying rule for module definition	34
12	Separate TTCN objects to be changed	34
13	Dependencies between Modules	35
14	The module size should be reasonable	35
15	Document the upper interface	36
16	Document the lower interface	36
17	ATS development from scratch	46
18	Document the import tables	47
19	Import without rewriting only in case of significant re-use	47
20	Re-validate modularised ATSs	48

Annex B: List of examples

To enable an easy and quick cross-reference to all examples that have been associated to rules in the body of the ETR, table B.1 below provides the complete list of the examples including the number of the example and the number of the page on which it appears:

Table B.1: List of examples

N^{o}	Example	Page
1	Using Basic Access or Primary Rate Access in ISDN	24
2	Same service provided by different protocols	24
3	X.25 protocol providing different services	25
4	Upper layer protocols providing service to different protocols.	25
5	Direction dependencies in SCCP connection oriented protocol.	25
6	Embedding of protocols in SS#7 stacks	25
7	ISDN D-channel controlling the usage of the B channel	25
8	Direct interaction	25
9	SCCP versions (1988 then 1992), GSM MAP phase1 then phase2	26
10	For X.400 a large quantity of profiles exist.	26
11	Many occurrences of the X.25 packet layer protocol in profiles	26
12	Unsolicited transmission of data PDUs	28
13	Upper tester treated as a test component	29
14	Typical alternative in X.25 layer 2	29
15	Usage of test step parameters to handle constraint field modification	31
16	A typical Exit test step when used in single-party context	33
17	A typical Exit test step when used in multi-party context	33
18	Documentation of Modules interfaces	36
19	National ISDN ATSs	44

Annex C: Unresolved issues on TTCN Modules

While modularity in TTCN is still under discussion, some rules and examples referring to TTCN Modules have been stated conditionally in this ETR. These conditions reflect the different positions expressed in several proposals to extend TTCN to provide the feature of TTCN Modules.

The conditions are related to some options in the definition of TTCN Modules, that might be finally realised in ISO/IEC 9646-3 [3], or not. For central reference, the most important open possibilities are listed in this annex:

a) Allow the use of **external objects** in a module.

This gives a broad flexibility in defining TTCN Modules, but it restricts the possible semantics checks that can be made for an isolated module.

b) Allow import **from test suites**, not only from Modules.

This feature is particularly useful for application on existing test suites, which were created before the module concept was defined. It is also useful for the adaptation of test suites to a profile-specific context.

c) Allow the **override** (replace) feature.

This feature allows that some imported objects are overridden or replaced by objects **explicitly** defined in the importing module or test suite. The use of this feature makes only sense when the overridden objects are referred to by some other imported object (e.g. a test case attaches a test step), and a redefinition is necessary for the overridden objects (in the context of the importing module or test suite).

NOTE: Some proposers would also allow to override an imported object by another imported object. But this introduces a preference or order on the Modules from which is imported, which could easily lead to conflicts and circular dependencies.

d) Allow the import of **comprehensive collections of TTCN objects** (e.g. the Constraints Part) in one line of the import table, not only individual named TTCN objects.

This feature can make import tables small, and is particularly useful when the importing test suite or module contains only a few additional explicit definitions. This is likely to happen, for example, in a PTS, where a few additional test cases (and possibly constraints etc.) have to be defined with respect to a given ATS used in the PTS.

This feature can also decrease the readability, because the single imported objects are "hidden". The application is restricted, when because of naming conflicts single objects have to be renamed. Unlike in a), b) and c), no new function is introduced, but more "convenience".

History

Document history	
August 1995	First Edition
February 1996	Converted into Adobe Acrobat Portable Document Format (PDF)

ISBN 2-7437-0228-1
Dépôt légal : Août 1995