Chapter 1

TOWARDS THE THIRD EDITION OF TTCN

Jens Grabowski and Dieter Hogrefe
Institute for Telematics, University of Lübeck,
Ratzeburger Allee 160, D-23538 Lübeck, Germany
{jens,hogrefe}@itm.mu-luebeck.de

Abstract

The third edition of TTCN (Tree and Tabular Combined Notation) will be a complete redesign of the entire test specification language. The close relation between graphical and textual representation will be removed, OSI specific language constructs will be cleared away and new concepts will be introduced. The intention of this redesign is to modernize TTCN and to widen its application area beyond pure OSI conformance testing. This paper motivates the need for a new TTCN, explains the design principles and describes the status of the work on the third edition of TTCN.

Keywords:

ETSI, ITU-T, CORBA, OSI Conformance Testing, Test Specification, TTCN, Programming Languages

1. INTRODUCTION

The *Tree and Tabular Combined Notation* (TTCN) is a well established notation for the specification of test cases for OSI protocol conformance testing. TTCN is defined and standardized in Part 3 of the international standard 9646 'OSI Conformance Testing Methodology and Framework' (CTMF) [2].

OSI conformance testing is understood as functional black-box testing, i.e., a *system under test* (SUT) is given as a black-box and its functional behavior is defined in terms of inputs to and corresponding outputs from the SUT. Subsequently, TTCN test cases describe sequences of stimuli to and required responses from the SUT.

CTMF and TTCN have been used for testing OSI protocols and protocols in systems following the OSI layering scheme, e.g., ISDN or ATM. CTMF principles and TTCN have also been applied successfully to other types of functional black-box testing, e.g., ISDN service testing and interoperability testing.

The requirements on testing are changing. The testing of new software architectures, e.g., ODP, CORBA, TINA or DCE, with advanced and time-critical applications, like, multimedia, home-banking, or videoconferencing, requires new testing concepts, new testing architectures and a new and powerful test specification language.

Researchers have already started to extend CTMF and TTCN in order to meet the upcoming testing requirements. The proposed extensions were related to testing architectures [5, 9], real-time testing [7, 8] and performance testing [6]. Some of these ideas will find their way into practice.

The international standardization organizations *International Telecommunication Union* (ITU-T) and *European Telecommunications Standards Institute* (ETSI) have studied the new testing requirements [1] and recognized the urgent need for a modern and powerful test specification language. As a consequence, the *specialists task force* (STF) 133 was set up in October 1998. STF 133 will (1) correct the known defects in the second edition of TTCN and (2) develop the third edition of TTCN (TTCN-3) which is an extension and complete redesign of the previous TTCN editions.

The correction of the known defects has been finished in December 1998 and the revised second edition of TTCN will be published by ETSI in 1999. The development of TTCN-3 is an ongoing task and is expected to be completed in spring 2000. This paper introduces TTCN-3 and describes the current status of the work on TTCN-3.

2. REQUIREMENTS ON TEST LANGUAGES

A test specification language should fulfil some general requirements which distinguish it from other specification or programming languages. In this section, these requirements are listed and compared with the properties of TTCN.

2.1 GENERAL REQUIREMENTS

A test specification language should

1. provide functionality which is specific to testing and which cannot be easily made available in other programming or specification languages,
2. allow to present the details of the implemented test purpose in a human understandable form,
3. be transferable between different computers,
4. be compilable and afterwards be executable on some test equipment,
5. be versatile and not only be suited for one application area, and
6. be easy to learn and understand.

This list is very general and the requirements may not be equally important. However, it is obvious that the fulfillment of each of these requirements will improve the acceptance of a test specification language.

2.2 NEGATIVE PROPERTIES OF TTCN

TTCN provides a graphical form (TTCN/GR) and a textual machine processible form (TTCN/MP). The graphical form is table oriented, i.e., a TTCN/GR test suite is a collection of different kinds of tables. TTCN/GR and TTCN/MP are closely related. In fact, there is a one-to-one mapping between each row in a TTCN/GR table and a line in a TTCN/MP file.

TTCN is too restrictive. The first negative property of TTCN which is valid for TTCN/GR and TTCN/MP is that TTCN is too restrictive. The browser structure of a TTCN test suite is built into the syntax and provides an OSI conformance testing oriented view. In addition, TTCN includes several concepts and application-specific static semantic rules which have only a meaning in OSI conformance testing. For example, the distinction between upper and lower tester functions or the distinction between PDUs and ASPs. In addition, a compiler will treat some conformance testing specific information only as comments.

The proforma format of TTCN/GR is also too restrictive. Reordering of information to improve readability is not possible. Omitting rows or columns which are not needed is not possible either. In case of long table entries, the column form is not always suitable. Using TTCN/MP in such cases does not help, because TTCN/MP reflects the table structure and cannot be used like a normal programming language.

As a summary, it can be stated that TTCN violates the requirements on readability (2, 6) and versatility (5). An enhanced TTCN should provide views beyond OSI conformance testing, should provide a human readable (usable) textual form and may support other presentation forms than pure tables, e.g., MSC, SDL or Java-like.

TTCN is too complex. The second negative property of TTCN is its complexity. The grammar rules of TTCN/MP are too complex and TTCN/GR includes too many different proformas. In total, TTCN/GR distinguishes 47 different types of tables (without compact proformas) where some only differ in the table headings. This makes it difficult to learn and read TTCN/GR test suites.

TTCN includes some redundant functionality, e.g., macros versus structs or compact proformas. Such functionality makes a language clumsy, complex and reduces its readability and usability. Typically, it remains in the language definition due to backwards compatibility.

Some TTCN complexity is caused by language constructs supporting functionality which normally should be provided by a tool. For example, the index of test cases and test steps with page numbers is something which should be provided by TTCN tools but should not be part of the language itself.

The conclusion of this discussion is that TTCN violates the requirements on readability and simplicity (2, 5). A new test specification language should be simple, less complex and should not support functionality which can be provided by a tool.

2.3 POSITIVE PROPERTIES OF TTCN

In spite of all the negative properties, TTCN has a lot of positive properties too. TTCN provides functionalitied and concepts which are specific to testing and cannot be provided easily by other languages (requirement 1). These functionalities and concepts are related to

- test case selection and test case parameterization,
- the concept of *points of control and observation* (PCOs),
- test configurations,
- the link to ASN.1,
- the inclusion of encoding information,
- the concept of constraints,
- matching mechanisms,

- verdict assignment, and
- the specific operational semantics (snapshot-semantics)

Beyond that, TTCN has proven to be transferable between different computers, and to be compilable and executable (requirement 3, 4).

2.4 APPLICATION REQUIREMENTS

TTCN was developed to support the conformance testing procedure of OSI protocol implementations. To meet the testing requirements of new software architectures and advanced applications, the enhanced TTCN has to include new concepts [1]. These are:

- options to specify dynamic test configurations,
- support of additional communication mechanisms, e.g., synchronous communication and broadcast communication,
- an extended timer concept to allow the test of hard real-time requirements
- support for the specification of performance tests.

3. PRINCIPLES OF TTCN-3

As a consequence of the discussions about the requirements on test specification languages and the properties of TTCN, the ETSI *technical committee* (TC) *methods for testing and specification* (MTS) established STF 133 in its funded work program. Beyond the already mentioned and finished correction of the second edition of TTCN, STF 133 will develop TTCN-3 with the following goals in mind:

- simplification of TTCN,
- harmonization of the latest editions of ASN.1 [3] and TTCN,
- integration of new communication concepts such as synchronous communication and monitoring, and
- support of dynamic test configurations in TTCN.

It was decided to base the work of STF 133 on the existing experience with TTCN and CTMF. Due to restricted resources, concepts for real-time and performance testing will not be included in TTCN-3. The goals listed above will be reached by a complete redesign of the existing TTCN. The work will concentrate on the development of a textual syntax with a look-and-feel similar to a programming language.

TTCN-3 may have several graphical representation formats. The ETSI representation format will be defined by STF 133. It will be table-oriented and based on 15 generic proformas. For example, there will be only one generic proforma for the definition of ASPs, PDUs and coordination messages (CMs).

4. TTCN-3 - STATE OF WORK

At the time of writing this paper, only a few TTCN-3 language issues have been investigated thoroughly and only a few syntax proposals have been made. To give an impression of TTCN-3, the harmonization of ASN.1 and TTCN-3, the concepts for modularization and some ideas about behavior descriptions in TTCN-3 are discussed.

4.1 ASN.1 AND TTCN-3 HARMONIZATION

The harmonization of ASN.1 and TTCN-3 is done to allow the combined use of the latest version of ASN.1 and TTCN-3. It is a difficult task, because ASN.1 and TTCN-3 are distinct languages with different syntax and semantics definitions. In the previous versions, the combined use of ASN.1 and TTCN has led to a mixture of syntax rules. In some cases, it was allowed to use TTCN language constructs in ASN.1 descriptions, e.g., TTCN matching mechanisms in a ASN.1 value notation, and vice versa. A compiler has to decide from the context whether syntax rules of ASN.1, TTCN, or both have to be applied.

The TTCN-3 strategy to avoid such problems is the separation of ASN.1 and TTCN-3 constructs. ASN.1 definitions have to be placed in a ASN.1 block (Figure 1.1) and all definitions in such a block follow the ASN.1 syntax rules. The definitions can only be used by reference and it is not allowed to use ASN.1 syntax in TTCN-3 constructs.

In some cases, this scheme may lead to minor problems, e.g., the use of matching mechanisms in and the modification of constraints which refer to ASN.1. In this case, the solution is to transform such constraints into the TTCN table notation which can be modified according to the TTCN rules.

The advantages of this harmonization approach seem to be bigger than the recognized problems. In addition, this solution may open a door for TTCN-3 to application areas where other data descriptions are used, e.g., IDL in the CORBA context [4].

4.2 MODULARIZATION

TTCN-3 will only have modules and will not distinguish between test suites and modules. The term *modularization* in TTCN-3 refers to the

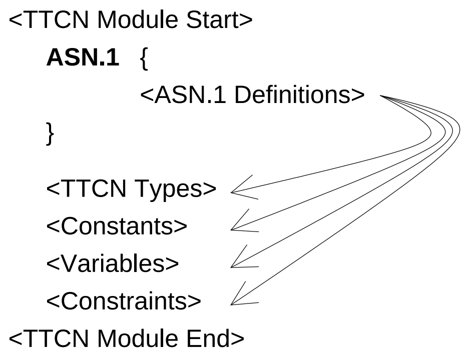


Figure 1.1 ASN.1 block in TTCN-3

possibility of reusing the definitions of a module in another module. For this, the definitions have to be imported by the latter one.

TTCN-3 will only have an `import` construct but no `export` construct. All global definitions of a module may be imported by another module.

The names in a module have to be unique. Name conflicts due to the `import` from different modules have to be resolved by implicit and explicit prefixing with the identifier of the module from which the name is imported.

The `import` construct should be easy to use and avoid the writing of long import lists. Additional keywords will help to keep the `import` construct user friendly. Two examples may illustrate this. The statement

```
import typedef MyType from MyModuleC recursively;
```

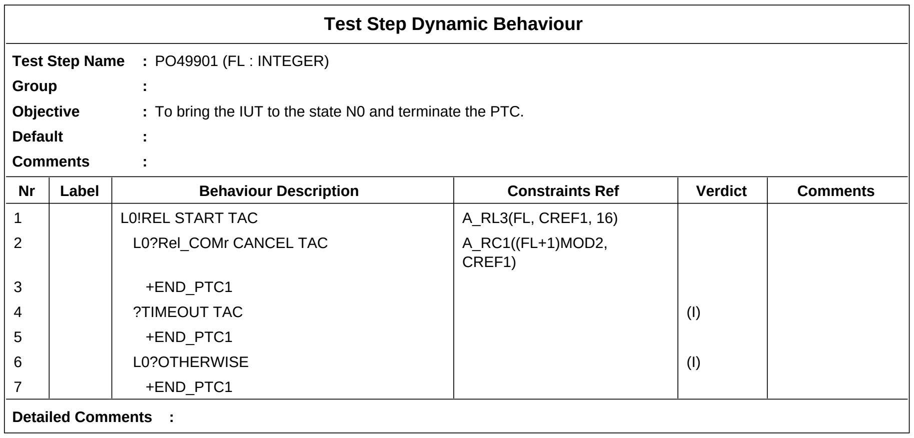
describes the import of the type definition `MyType` from `MyModuleC`. The keyword `recursively` states that all type definitions which may be used within `MyType` are also imported. The statement

```
import all from MyModuleD exclude all constraints;
```

denotes the import of all definitions except for its constraints from `MyModuleD`.

4.3 BEHAVIOR DESCRIPTIONS

Behavior will be introduced in TTCN-3 on the level of modules, on the level of module operations and on the level of test cases. Behavior on the level of modules allows to specify test control programs. For example, a test case is executed only if another test case has been executed successfully, or a test case is repeated several times. Behavior on the level of module operations corresponds to test suite operations in the second edition of TTCN. Behavior on the level of test cases corresponds to the behavior descriptions of test cases, test steps and defaults in the



Test Step Dynamic Behaviour					
Test Step Name : PO49901 (FL : INTEGER)					
Group :					
Objective : To bring the IUT to the state N0 and terminate the PTC.					
Default :					
Comments :					
Nr	**Label**	**Behaviour Description**	**Constraints Ref**	**Verdict**	**Comments**
1		L0!REL START TAC	A_RL3(FL, CREF1, 16)		
2		L0?Rel_COMr CANCEL TAC	A_RC1((FL+1)MOD2, CREF1)		
3		+END_PTC1			
4		?TIMEOUT TAC		(I)	
5		+END_PTC1			
6		L0?OTHERWISE		(I)	
7		+END_PTC1			
Detailed Comments :					

Figure 1.2 A TTCN test step dynamic behavior description

second edition of TTCN. STF 133 will select the constructs for describing the flow of control (loops, conditions, jumps) in such a way that they can be used on all levels.

On the level of test cases TTCN-3 will distinguish between behavior definitions and the invokation of an instance of a behavior definition. An instance of a behavior definition corresponds to a test case which will be executed when the module is applied to an SUT. A behavior definition may call other behavior definitions for describing more complex behavior and it may be instantiated. This means, it depends on the use of a behavior definition whether it can be interpreted as a test case, test step or default behavior description. Behavior definitions can be imported by other TTCN-3 modules. A behavior definition can be instantiated by reference, i.e., the name of the behavior definition is referenced, or in form of an inline definition, i.e., the behavior definition is provided in the place of its instantiation.

The following example may provide an idea of the TTCN-3 'look-and-feel' in comparison with the second edition of TTCN. In the ETSI abstract test suite for the ISDN supplementary service Multiple Subscriber Number (ETSI EN 300 052 6) the test step definition shown in Figure 1.2 can be found (for simplicity, the comments and group reference are left out). The corresponding TTCN-3 representation is shown in Figure 1.3.

In Figure 1.2, ASP or PDU type references can be found on behavior lines, timer commands are related to send and receive events, the verdict assignment is done on behavior lines and the test step `END_PTC1` has to be attached three times, i.e., to all three branches of the tree.

```
behaviour P049901(FL integer)
/* Objective:  To bring the IUT
to the state N0 and
terminate the PTC1.  */
{
  L0!A_RL3(FL,CREF1,16);
  start(TAC);
  alt {
    A1: L0?A_RC1((FL+1) mod 2, CREF1);
        cancel(TAC);
    A2: ?timeout(TAC);
        (inconclusive);
    A3: ?otherwise;
        (inconclusive);
  };
  +END_PTC1;
}
```

Figure 1.3 TTCN-3 representation of the test step in Figure 1.2

In Figure 1.3, the statements are ordered sequentially. The sequence starts with a send statement, followed by a start timer, followed by an alternative (which for the first alternative includes a sequence of two statements). The behavior description ends with the attachment of `END_PTC1`. Behavior lines in Figure 1.2 describing several actions are translated into separate commands (e.g., the first line in Figure 1.2 `L0!REL START TAC` is translated into `L0!A_RL3(FL,CREF1,16); START TAC;`). Verdict assignment is done by special commands which can be used anywhere in the behavior definition. Only constraints are referenced by send and receive events. The ASP or PDU type references found in the second edition of TTCN are superfluous because they are provided within the constraint definitions.

5. SUMMARY AND OUTLOOK

In this paper, the motivation for the development of TTCN-3 has been discussed and the status of the work has been described. TTCN-3 will be a completely new test specification language which widens the application area of TTCN beyond pure OSI conformance testing as defined in CTMF. This new test specification language will have a look-and-feel like a normal programming language but will keep the testing specific properties of the previous versions of TTCN. TTCN-3 will have a standardized textual syntax, but may have several graphical representation formats. The ETSI representation format will be table-oriented and include 15 different generic proformas. At the time of writing this paper, the development of TTCN-3 was in a state where only a few issues have been studied thoroughly. Nevertheless, it is planned to finish the development of TTCN-3 and the ETSI presentation format in spring 2000.

Acknowledgments

The work presented in this paper has been carried out within STF 133 as part of the funded work program of ETSI. The work on dynamic test configurations within STF 133 is funded by Nortel. One of the authors is a member of STF 133. The authors would like to thank Anthony Wiles, Colin Wilcock and Guy Martin for allowing us to present parts of the STF 133 work.

References

[1] ETSI TC MTS. *Guide for the use of the second edition of TTCN (Revised Version).* European Guide 202 103, 1998.

[2] International Standardization Organization. *Information Technology — OSI — Conformance Testing Methodology and Framework — Parts 1–7.* ISO, International Standard 9646, 1994 - 1997.

[3] ITU-T Recommendations X.680-683. *Information Technology - Abstract Syntax Notation One (ASN.1)*, 1994.

[4] Object Management Group. *The Common Object Request Broker: Architecture and Specification; Revision 2.0.* Frammingham, Massachusetts, USA, 1995.

[5] I. Schieferdecker, M. Li, A. Hoffmann. *Conformance Testing of TINA Service Components - the TTCN/CORBA Gateway.* In Proceedings of the 5th International Conference on Intelligence in Services and Networks, Antwerp, Belgium, May 1998.

[6] I. Schieferdecker, S. Stepien, A. Rennoch. *PerfTTCN, a TTCN Language Extension for Performance Testing.* In M. Kim, S. Kang, K. Hong, editors, *Testing of Communicating Systems*, volume 10, Chapman & Hall, September 1997.

[7] T. Walter, J. Grabowski. *Real-time TTCN for Testing Real-time and Multimedia Systems.* In M. Kim, S. Kang, K. Hong, editors, *Testing of Communicating Systems*, volume 10, Chapman & Hall, September 1997.

[8] T. Walter, J. Grabowski. *A Framework for the Specification of Test Cases for Real-Time Systems.* Accepted for Publication in:

Journal of I nformation and Software Technology, Special Issue on Communications Software Engineering, 1999.

[9] T. Walter, I. Schieferdecker, J. Grabowski. *Test Architectures for Distributed Systems - State of the Art and Beyond* (Invited Paper). In A. Petrenko, N. Yevtushenko, editors, *Testing of Communicating Systems*, volume 11, Chapman & Hall, September 1998.

6. BIOGRAPHY

Jens Grabowski studied computer science and chemistry at the University of Hamburg, Germany, where he graduated with a diploma degree. From 1990 to October 1995 he was a research scientist at the University of Berne, Switzerland, where he received his Ph.D. degree in 1994. Since October 1995 Jens Grabowski is a researcher and lecturer at the Institute for Telematics. He is a member of the ETSI specialists task force 133 which develops the third edition of TTCN.

Dieter Hogrefe studied computer science and mathematics at the University of Hannover, Germany, where he graduated with a diploma degree and later received his PhD. From 1983 to 1995 he worked at various positions in the industry and at universities. Since 1996 he is director of the Institute for Telematics and full professor at the Medical University of Lübeck. Dieter Hogrefe is also chairman of ETSI TC MTS.