Development of an MSC/UML Test Format

Ekkart Rudolph[1], Ina Schieferdecker[2], Jens Grabowski[3]

[1]Technische Universität München, Institut für Informatik, D-80290 München
Email: rudolphe@informatik.tu-muenchen.de

[2]GMD FOKUS, Kaiserin-Augusta-Allee 31, D-10589 Berlin,
Email: Schieferdecker@fokus.gmd.de

[3]Medizinische Universität zu Lübeck. Institut für Telematik, Ratzeburger Allee 160,
D-23538 Lübeck, Email: grabowsk@itm.mu-luebeck.de

Abstract

The development of an Message Sequence Chart (MSC) based graphical representation of the Tree and Tabular Combined Notation (TTCN) is part of the ETSI project STF 156 on 'Specification of a Message Sequence Chart/UML format, including validation for TTCN-3'. Experiments with different kinds of MSC representations show that certain extensions of the MSC language are requested in order to obtain a sufficiently transparent and readable MSC test format. Extended HMSCs where reference symbols may either contain hypertext–like descriptions or, in the expanded form, the detailed event structure of basic MSCs appear to be especially suitable for a compact and transparent test case representation. For an effective usage of such advanced MSC constructs, a corresponding tool support appears to be mandatory where the event structure of special paths in the test case is described explicitly while others are hidden in MSC references containing pure textual descriptions. We propose the name 'HyperMSCs' for such extended HMSCs. A significant difference between the standard TTCN format and the MSC format refers to the behavior description of several test components. In the concurrent case, a TTCN test case describes the communication of the main test component (MTC) and its ports. The behavior description of parallel test components (PTCs) is provided by separate functions. A simple translation of this partitioning into MSCs leads to separate MSC specifications for the different test components which have to be merged somehow by an appropriate join operation. However, such a 'local' view does not make use of the main advantage of the MSC language. The strength of MSC lies in the immediate description of the communication behavior which is provided by a 'global' view including all test components in the same diagram. The translation of TTCN into the MSC format is demonstrated using a simple example taken from the Inres protocol.

1 Introduction

The third edition of the Tree and Tabular Combined Notation (TTCN)[1] [2,3,8] is a textual test specification language which looks like a common programming language, e.g., C or C++. As such, it allows the use of different graphical presentation (display) formats. Apart from the tabular (conformance testing) presentation format known from TTCN-2 [9], the development of a Message Sequence Chart (MSC) format appears to be of special interest and therefore is part of the ETSI project STF 156 [10].

[1] In the following the terms TTCN-2 and TTCN-3 are used to distinguish explicitly between the second and the third edition of TTCN.

The ITU-standard language MSC is a widespread and popular means for the visualization of system runs within (tele)communication systems [13]. A main advantage of the MSC language is its clear graphical layout which immediately gives an intuitive understanding of the described behavior. Within the area of conformance testing, MSC is already well established for the specification of test purposes and as such for the automatic generation of TTCN-2 test cases. Practice has shown that the tabular format of TTCN-2 is not very intuitive for behavior description even if tools are used. Using MSC as presentation format for TTCN-3 may considerably improve the readability of test cases and make them more understandable. At the same time, MSC in form of Sequence Diagrams forms a central constituent of UML and is employed for the formalization of Use Cases [6]. Therefore, an MSC format could bring the use of TTCN significantly closer to users of UML. Since there is no accepted test notation in UML, this is an ideal opportunity to bring TTCN-3 closer to the UML world. The current UML Sequence Diagrams cover only a subset of the expressive power of MSC. However, the ongoing work on UML v2.0 will also enhance the expressiveness of Sequence Diagrams.

MSC has its strong points in the behavior description. Therefore, the MSC presentation format for TTCN-3 will concentrate on the clear and intuitive presentation of test behavior. MSC can make use of any data type language to describe data dependencies. Therefore data types and test data will be presented by re-using TTCN-3 textual format or the tabular presentation format for TTCN-3.

Though MSC has been used for test specifications in the past only in a fairly restricted manner, the powerful composition mechanisms contained in the present version of the MSC language make a comprehensive MSC specification of test cases feasible. For that, both MSC inline expressions and High Level MSCs (HMSCs) [5,13] may be employed depending on the level of abstraction and the focus of representation. However, a naive translation of TTCN-3 into MSC would lead to overloaded diagrams which are difficult to read and to handle.

In the next chapter a short introduction to TTCN is given. In Chapter 3, it is demonstrated by means of an example taken from the Inres protocol [1] that in general, the naive translation of TTCN specifications into nested MSC inline expressions leads to diagrams which are less transparent than the TTCN representation. An appropriate MSC test format is developed by extending HMSCs to HyperMSCs. Within Chapter 4, a HyperMSC representation for concurrent TTCN is provided. Using the Inres test case as an example, it is demonstrated that the separate HyperMSCs obtained from a direct translation of concurrent TTCN may be merged to one combined HyperMSC describing the complete specification. Finally, in Chapter 5, the proposed MSC test format is discussed with respect to future elaboration.

2 TTCN

The TTCN is a semi-formal test notation that supports specification of abstract test suites for conformance testing [2,3]. An abstract test suite is a collection of abstract test cases. TTCN-2 is defined in Part 3 of the Conformance Testing Methodology and Framework for OSI-based systems (e.g., communication protocols and services) standardized by ISO/IEC and ITU [11]. TTCN-2 is the notation to express conformance testing concepts of this framework. The T for ‘tabular’ refers to the use of tables (proformas) for the graphical representation of test suites. The T for ‘tree’ refers to the hierarchical organization of a test suite as well as to the tree-like behavior of test cases. It has been proven that TTCN-2 is applicable in a much wider scope of applications than OSI protocols and services such as for conformance testing of ODP, TINA and CORBA [7], and IP-based systems, APIs, and reactive systems in general.

Currently, the third edition TTCN (TTCN-3) is worked out by ETSI [2,3,8]. TTCN-3 is a text-based language for the specification of tests for reactive systems. TTCN-3 is on syntactical (and methodological) level a drastic change to TTCN-2, however, the main concepts of TTCN-2 have been retained and improved and new concepts have been included, so that TTCN-3 will be applicable for a broader class of systems. New concepts are, e.g., a test execution control to describe relations between test cases such as sequences, repetitions and dependencies on test outcomes, dynamic concurrent test configurations and test behavior in asynchronous and synchronous communication environments. Improved concepts are, e.g., the integration of ASN.1 [12], the module and grouping concepts to improve the test suite structure, and the test component concepts to describe concurrent test setups.

The top-level unit of a TTCN-3 test suite is the module which can import definitions from other modules. A module consists of a definitions part and a control part. The definitions part of a module covers definitions, e.g., for test components, their communication interfaces (so called ports), type definitions, test data templates, functions, and test cases. The control part of a module calls the test cases and describes the test campaign. For this, control statements similar to statements in other programming languages (e.g., if-then-else and while loops) are supported. They can be used to specify the selection and execution order of individual test cases.

Test cases describe the probes during the test campaign, i.e., they specify the test behavior. One can express a variety of test relevant behavior within a test case such as the alternative reception of communication events, their interleaving and default behavior to cover, e.g., unexpected reactions from the tested systems. In addition to the automatic test verdict assignment, more powerful logging mechanisms, e.g., for a detailed tracing, are provided. An example of a TTCN-3 test case definition is shown in Figure1. It will be described in the next chapter.

In addition to the pure textual format, TTCN-3 will define at least two presentation formats: a tabular conformance testing presentation format that resembles the tabular form of TTCN-2 [9] and an MSC presentation format that supports the presentation but also development of TTCN-3 test cases on MSC level [10].

3 Development of a test notation in form of HyperMSC

In TTCN-2 test case descriptions, statements are written on successive lines with either successively incremented indentations to indicate subsequent statements or with equal indentations to indicate alternatives. In case of highly nested alternatives, such a notation is not very user friendly. In Figure 1 an example of a TTCN-3 test case is shown. The test case is the result of a direct translation of a TTCN-2 test case for the Inres protocol [1]. The indentation in Figure 1 follows the TTCN-2 rules and is not necessarily required. Instead of equal indentations, TTCN-3 has the alternative construct (indicated by the **alt** keyword) where the different alternatives start with square brackets. It should be noted, that TTCN-3 also allows other specification styles than the tree-style of TTCN-2, e.g., a sequential-style used in C or C++ programs. However, for the MSC representation of TTCN-3, the tree-style is the most complicated case and, therefore, we concentrate on its intuitive visualization.

The most obvious and straightforward way to represent TTCN test cases (as shown in Figure 2) by MSC diagrams would be to use inline operator expressions for alternatives, iterations etc [13]. Practice has shown that apart from simple cases such a 'naive' translation does not lead to diagrams which are easy to ready and understand (Figure 2). As a consequence, it would not be clear whether such an MSC format would mean a progress with respect to the traditional TTCN notation or even a step back. In particular, inline operator expressions obscure the message flow of the 'standard' cases (pass verdict) by mixture with alternative parts.

```
testcase mi_synch1 () runs on MTCType {
   ISAP1.send( ICONreq : Connection_Request );
   alt {
      [] MSAP2.receive( MDATind : Medium_Connection_Request ) {
            MSAP2.send( MDATreq : Medium_Connection_Confirmation );
            alt {
               [] ISAP1.receive ( ICONconf : Connection_Confirmation ) {
                     ISAP1.send ( IDATreq : Data_Request(TestSuitePar) );
                     alt {
                        [] MSAP2.receive ( MDATind : Medium_Data_Transfer ) {
                              MSAP2.send ( MDATreq : cmi_synch1 );
                              ISAP1.send ( IDISreq : Disconnection_request );
                              alt {
                                 [] ISAP1.receive (IDISind : Disconnection_Indication ) {
                                       MSAP2.receive(MDATind : Medium_Disconnection_Request );
                                       verdict.set(pass)
                                    }
                                 [] MSAP2.receive( MDATind : Medium_Disconnection_Request ) {
                                       ISAP1.receive( IDISind : Disconnection_Indication );
                                       verdict.set(pass)
                                    }
                                 [] MSAP2.receive ( MDATind : Medium_Data_Transfer );
                                       verdict.set(inconclusive) // medium data transfer
                              }                                  // repetition
                           }
                        [] ISAP1.receive( IDISind : Disconnection_Indication );
                              verdict.set(inconclusive) // connection failure
                     }
                  }
               [] MSAP2.receive( MDATind : Medium_Connection_Request );
                     verdict.set(inconclusive) // medium connection request repetition
               [] ISAP1.receive( IDISind : Disconnection_Indication );
                     verdict.set(inconclusive) // connection failure
            }
         }
      [] ISAP1.receive( IDISind : Disconnection_Indication );
            verdict.set(inconclusive) // connection failure
   }
   stop
} /* End testcase mi_synch1 */
```

Figure 1: Inres example of a TTCN-3 test case

Apart from the usage of inline expressions, there are some other features to be mentioned in Figure 2. In comparison with the tree and tabular notation, the comments are represented in form of MSC comments and the setting of test verdicts is described making use of conditions. The ports ISAP1, MSAP2 and the test component MTC are represented by instances. The test case in Figure 1 corresponds to the non-concurrent form of TTCN, i.e., there only exists one (main) test component during test execution.

An obvious drawback of this representation is the fact that the PASS-, and INCONC- cases all appear in the same way. Certainly, it would be advantageous to have a means to single out the normal (PASS) case. The MSC standard MSC-2000 contains several structuring mechanisms [13]. For this special purpose, the MSC reference mechanism seems to be tailored. In order to mark out the PASS-case, the other cases may be represented in form of MSC references. This notation has an immediate drawback: Without looking at the definitions of the referenced MSCs, there is no immediate information on the non-PASS cases. That means apart from the PASS-case, the representation of the test case is not very intuitive.

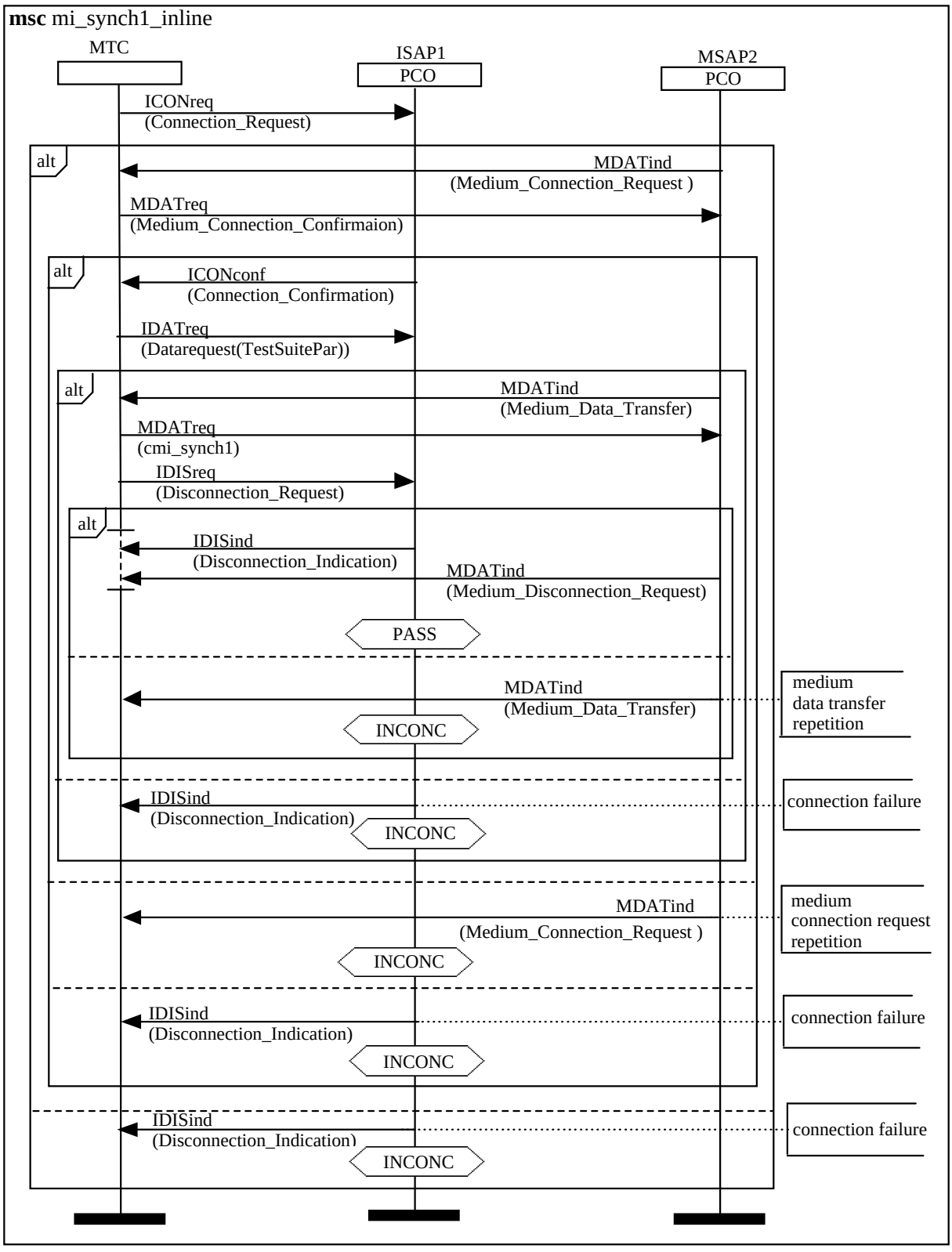

Figure 2: Representation of the Inres test case by means of inline expressions

We arrive at a much more satisfactory notation if we include a comment text together with the text verdict in the MSC references instead of reference names as is shown in Figure 3. It should be noted, however, that we deviate slightly from the MSC standard: The MSC references do not contain a name but an arbitrary text. In a sense, this can be viewed as a generalization of the MSC reference name convention, however, in practice it seems to be more appropriate to interpret this description in a hypertext–like manner: We assume a corresponding tool support where the MSC references can be expanded within the MSC in which they are contained or possibly also in a separate window. The MSC references which can be expanded may be indicated by thick lines or by colored or underlined text. Such an approach is appropriate particularly in the usual case of many fairly small MSC reference definitions. Because of this analogy, we have introduced the notation 'HyperMSC' which shall indicate not only a special syntax form but also the requirement for a corresponding tool support.

In case of many alternatives, the resulting representation is still not very transparent. As a general rule, inline expressions should be used only in very limited manner and should be restricted to only a few alternatives or loops. In more complex situations, HMSCs are much more transparent since they abstract from details and focus on the compositional structure [5,13]. However, if we translate the inline representation into an HMSC we are faced with the problem to represent the expanded parts since according to the standard MSC language an HMSC contains only non-expanded MSC reference symbols. In order, to overcome this deficiency we admit an expanded form of MSC references within HMSCs.

In comparison with the tree and tabular notation, the resulting HMSC still has the drawback, that the main event flow leading to a PASS verdict is split into separate parts. In case of many alternatives, this splitting is very disturbing. As a consequence, the MSC test format may not appear in every respect as a progress in comparison with the traditional TTCN representation. One would like to have a coherent expanded representation of a whole path not only in a separate window but also in inline-form within the HMSC itself. We therefore suggest a further extension of HMSCs which somehow may be viewed also as unification of HMSC and basic MSC. We combine the expanded MSC references to one coherent expanded MSC reference. As a consequence, we have to shift the branching point to the border line of the resulting MSC reference. The resulting HyperMSC is shown in Figure 4. The marked MSC references may be shown in expanded form. With a suitable tool support, the most attractive alternative representation would be to change the roles of the PASS-case, the INCONC-case and the FAIL-case if used.

4 HyperMSCs for concurrent TTCN

In the following, the concept of HyperMSC is demonstrated for the graphical representation of concurrent TTCN. The Inres test specification is represented in a concurrent manner. The test step mi_synch1_PTC_ISAP1 (Figure 5) contains the parallel test component PTC_ISAP1, the port ISAP1 which is used for the communication with the implementation under test and the port CP_ISAP1 which is used for the coordination of MTC and PTC. The test step mi_synch1_PTC_MSAP2 (Figure 6) contains the parallel test component PTC_MSAP2, the port MSAP2 and the (coordination) port CP_MSAP2. The test case mi_synch1 (Figure 7) contains the MTC which creates the PTCs. In addition, it only contains ports used for the coordination among the test components.

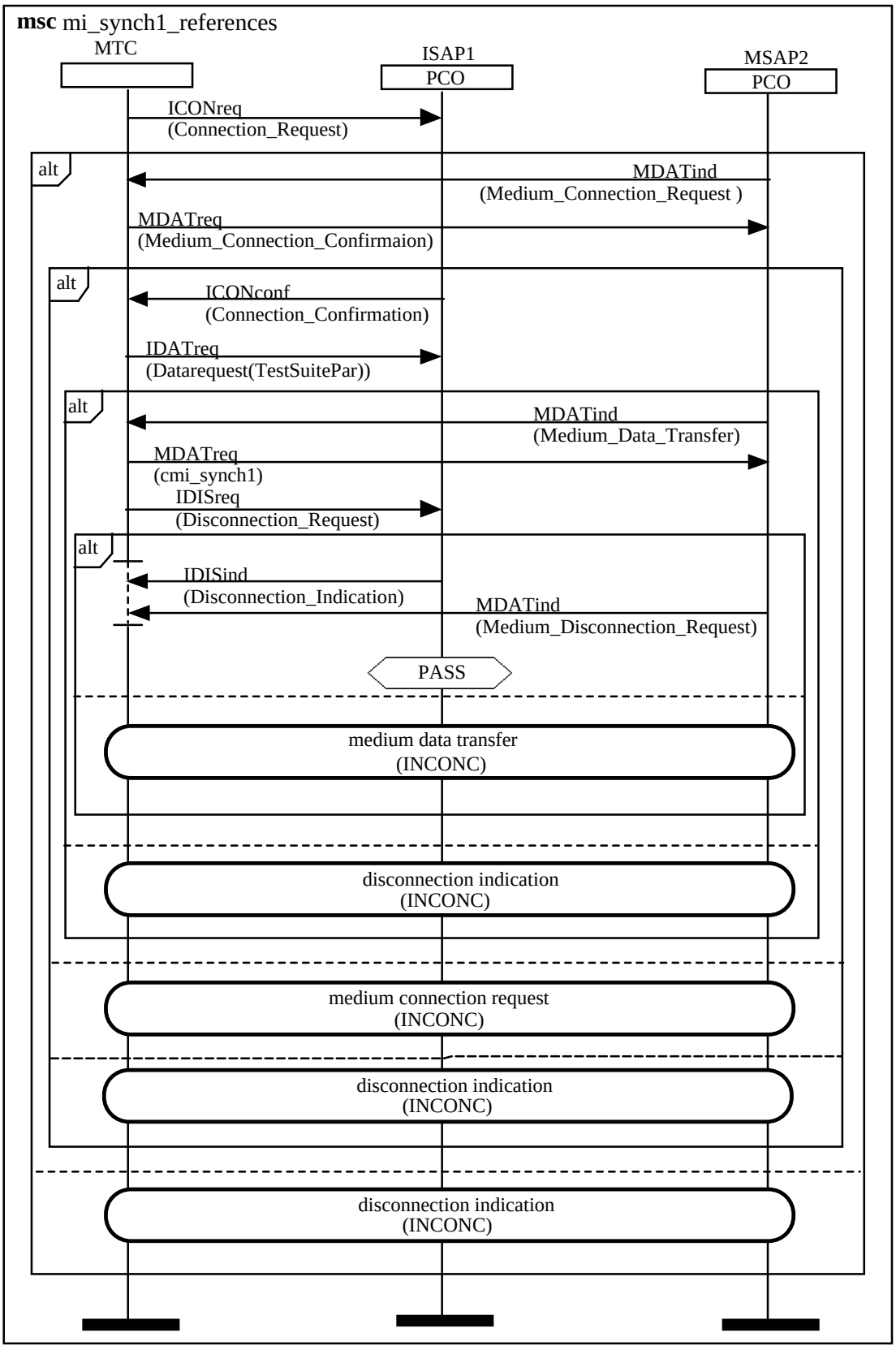

Figure 3: Representation of the Inres test case in form of an HyperMSC using inline expressions and MSC references with textual inscriptions

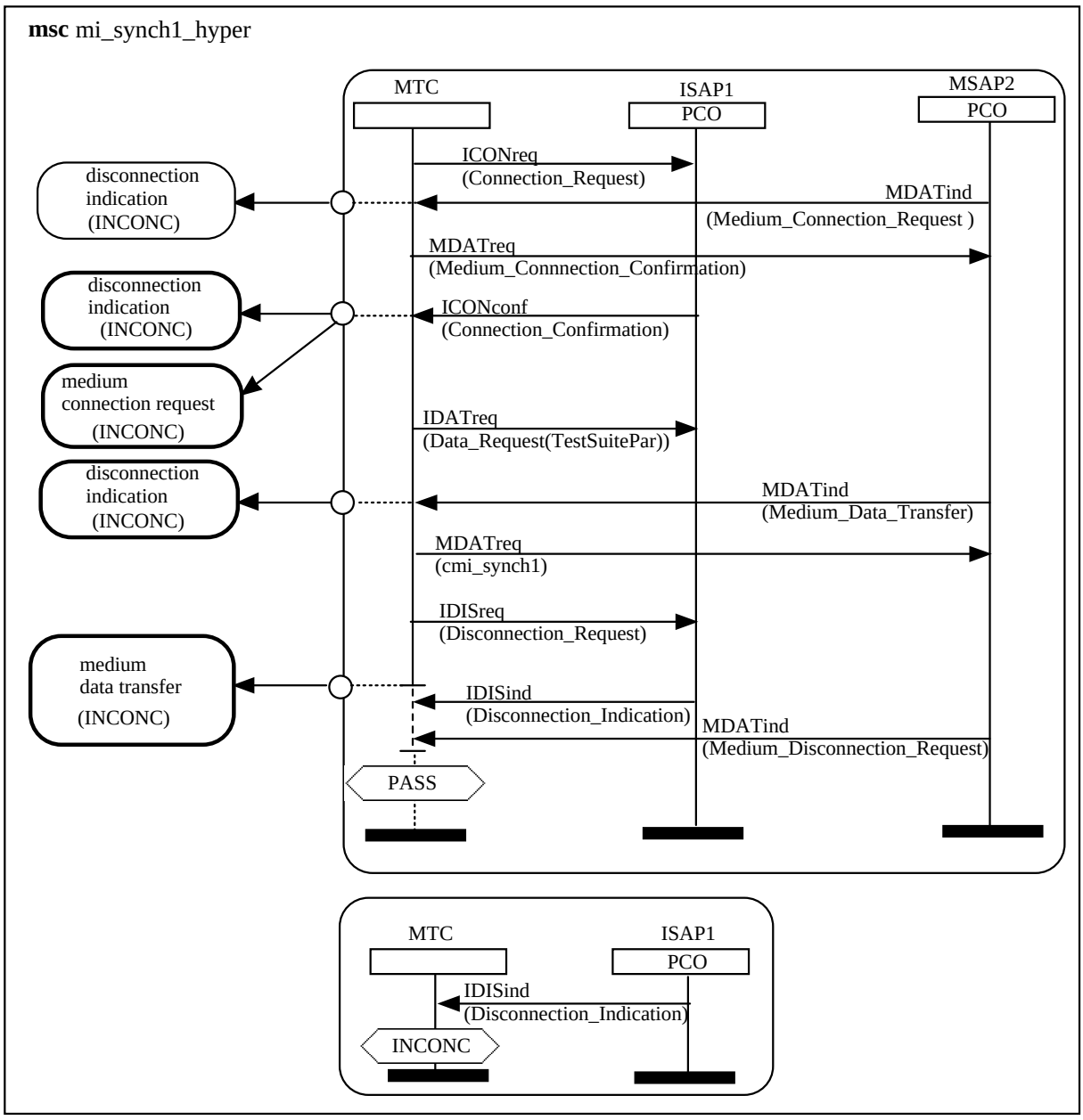

Figure 4: Representation of the Inres test case in form of an HyperMSC with generalized graphical branching construct. Expansion of an MSC reference in a separate representation

The three resulting HyperMSCs have to be merged via the coordination messages by means of a special *join* operation. Although such a representation is appropriate for many purposes, the main advantage of the MSC language in comparison with TTCN, SDL etc., lies in the explicit presentation of the communication behavior. In Figure 8, the MSCs of Figure 5-7 are merged explicitly within one embedding MSC INRES. Such a combined HyperMSC representation of concurrent TTCN is far from trivial and certainly needs further investigations. The MSC INRES contains three parallel HyperMSCs which are joined via the coordination messages. Note, that we have omitted the redundant rectangular frame of parallel composition for convenience.

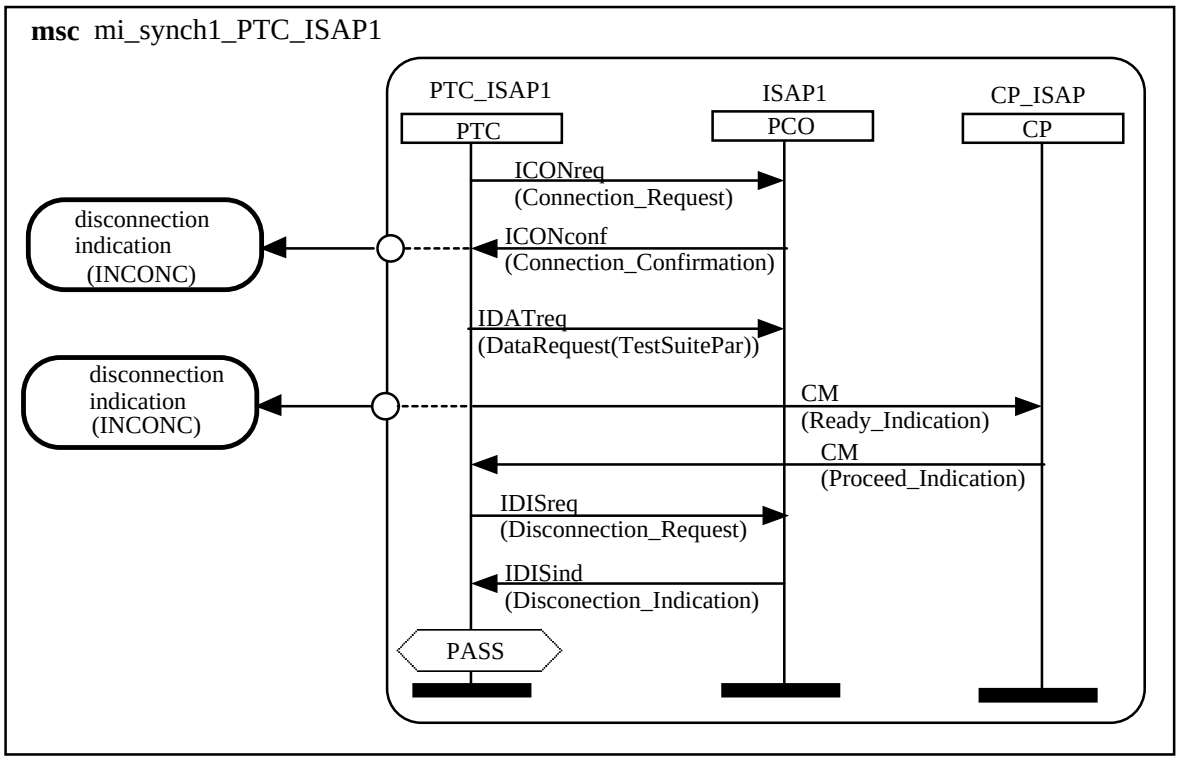

Figure 5: Hyper MSC representation for test step mi_synch1_PTC_ISAP1

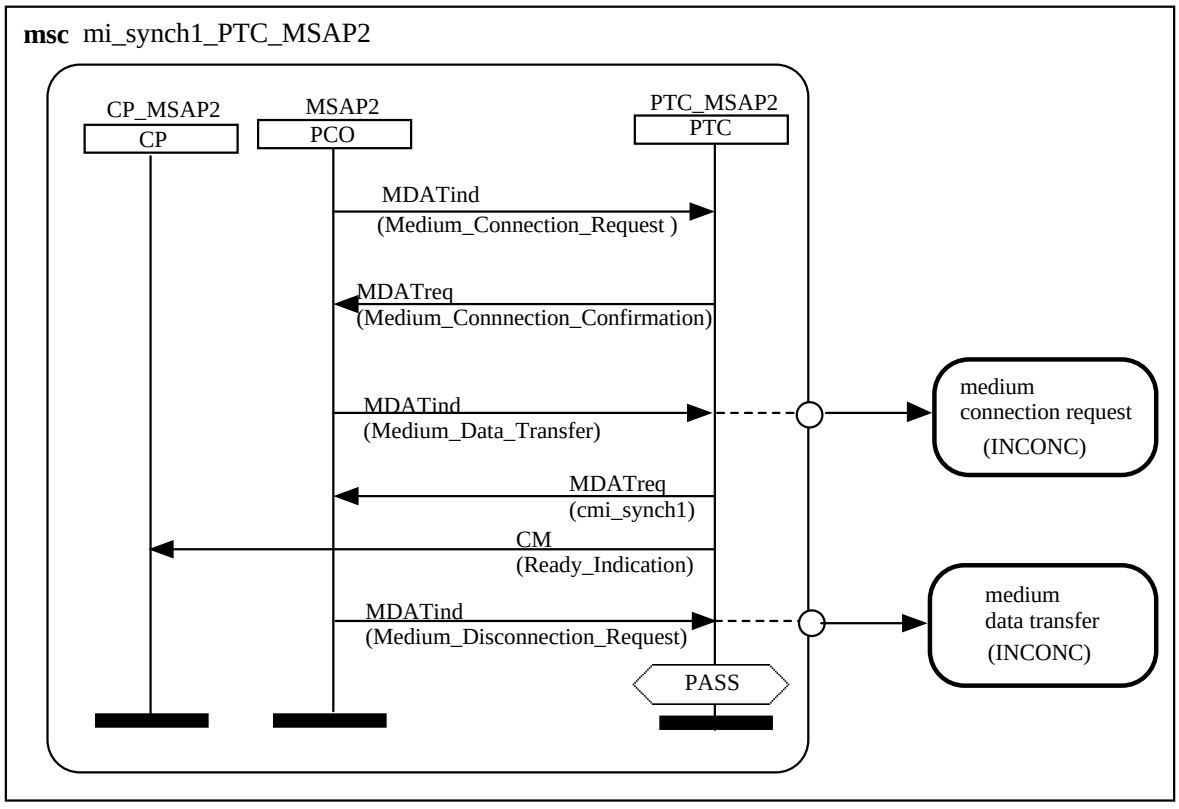

Figure 6: HyperMSC representation for test step mi_synch1_PTC_MSAP2

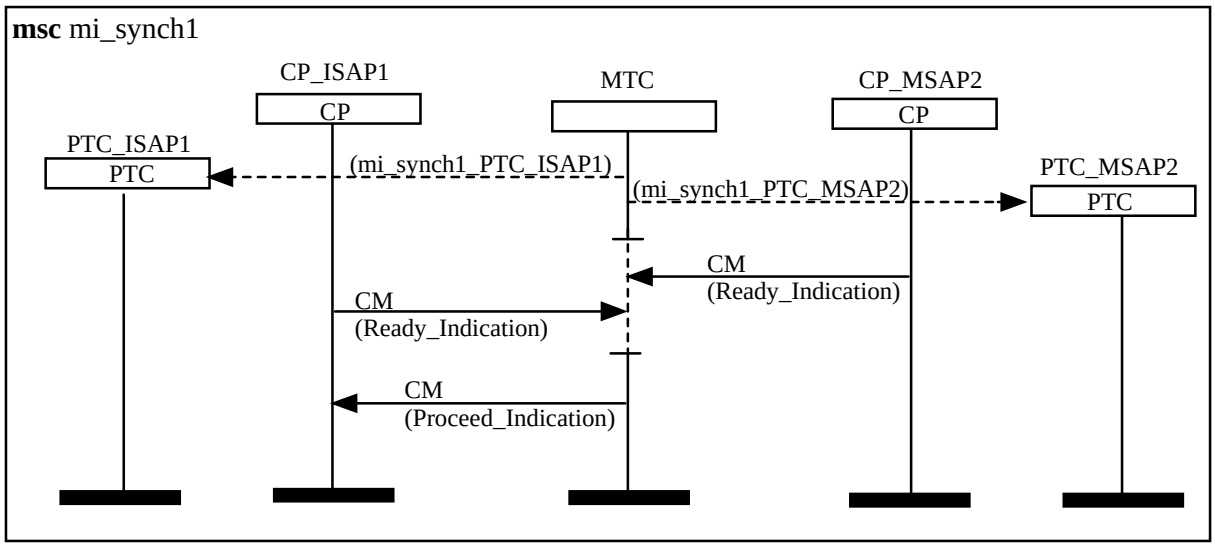

Figure 7: HyperMSC representation for test case mi_synch1

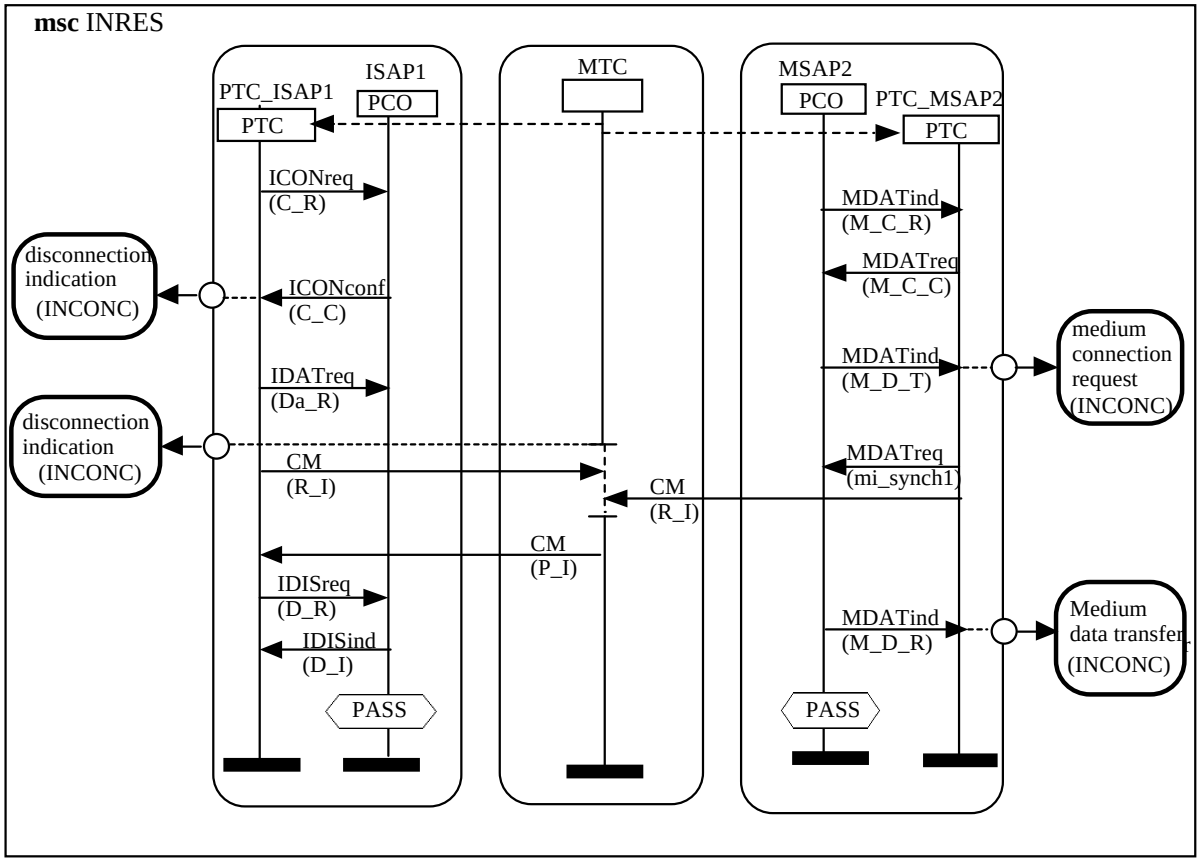

Figure 8: Merged representation by means of a HyperMSC

5 Conclusion and outlook

Obviously, the MSC standard needs certain extensions in order to be applicable for a graphical representation of TTCN test cases. Neither MSC inline expressions nor the standard HMSCs can be immediately employed for the development of an MSC test format which is sufficiently transparent and readable. Though the tree-like specification-style of test cases is far from being ideal it at least describes the normal cases in a coherent manner. An MSC test format which really can be looked at as a progress with respect to the tree-like representation demands a different handling of HMSCs together with an appropriate tool support. The proposed new features leading to the concept of 'HyperMSC' merely concern the layout and the handling but do not impose any substantial changes. In particular, there are no semantics changes requested. Beyond that, most of the changes like expanding MSC references within HMSCs or generalizing MSC reference name conventions to comment texts can be looked at as variants which are already employed in practice. That shows that the strict border line between HMSCs and plain MSCs is of no practical value and should be removed. The proposal of HyperMSC, therefore, can be looked at also as a unification of the MSC language.

The HyperMSCs in Chapter 3 and 4 exhibit the main paths (PASS-cases) in a coherent way in form of an expanded MSC diagram while the side cases (INCONC-cases) are indicated by branching with non-expanded MSC references containing textual descriptions. This suggests another attractive HyperMSC application which immediately combines the MSC format with the TTCN representation: The MSC references describing the side cases may contain even TTCN language descriptions instead of comments. Such a hybrid representation may be particularly useful for documentation purposes since it provides a complete test case description with a visualization of the main paths in form of the MSC format. As a consequence, even more than two levels of MSC reference representations may be distinguished. There are four possible ways to present a reference: (a) by its proper name, (b) by a textual description (comments and text verdict), (c) by means of TTCN program descriptions, (d) by its full blown MSC diagram.

Although the motivation for the development of HyperMSC was the development of a MSC test format within ETSI its possible application range is much larger. In particular, HyperMSC may be useful for the modeling and formalization of Use Cases by means of HMSCs.

The presented proposal for an MSC test format appears to be a major step towards an intuitive and user-friendly graphical representation of TTCN. However, there are still several open questions in the development of the MSC test format. While most of the main TTCN language constructs - communication operations, program statements, functions - allow a straightforward translation into the MSC language, for some TTCN constructs, e.g., the default construct or the test configuration, there is no immediate MSC counterpart. MSC-2000 contains a data part, however, it is not clear how it maps onto the data descriptions in test suites. The most convenient way probably is to take over the module definition part from TTCN and to translate only the module control part using HMSCs. At the same time, the idea of hypertext-like descriptions can be suitably carried over to the constraint references in attached to message names in parenthesis thus providing a convenient link to the definition part. The elaboration of these concepts is part of the ETSI project STF 156.

References

[1] F. Belina, D. Hogrefe, A. Sarma. *SDL with Applications from Protocol Specification*. Prentice Hall, 1991

[2] Jens Grabowski. *TTCN-3 - A new Test Specification Language for Black-Box Testing of Distributed Systems*. Proceedings of the 17th International Conference and Exposition on Testing Computer Software (TCS'2000), Theme: Testing Technology vs.Testers' Requirements, Washington D.C., June 2000.

[3] J. Grabowski, D. Hogrefe. *An Introduction to TTCN-3*. In: G. Csopaki, S. Dibuz, K. Tarnay, editors, , 'Testing of Communicating Systems – Methods and Applications', Kluwer Academic Publishers, September 1999.

[4] J. Grabowski, T. Walter. *Visualisation of TTCN test cases by MSCs*. In: Proceedings of the 1st Workshop of the SDL Forum Society on SDL and MSC - SAM'98 (editors: Y. Lahav, A. Wolisz, J. Fischer, E. Holz), Informatik-Berichte Humboldt-Universität zu Berlin, June 1998.

[5] S. Mauw, M. A. Reniers. *High Level Message Sequence Charts*. In SDL'97 Time for Testing-SDL, MSC and Trends, Proceedings of the 8th SDL Forum in Evry France (A. Cavalli and A. Sarma editors), North Holland, Sept. 1997.

[6] E. Rudolph, J. Grabowski, P. Graubmann. *Towards a Harmonization of UML Sequence Diagrams and MSC*. In SDL'99, The Next Millennium, Proceedings of the 9th SDL Forum in Montreal Canada (Yair Lahav and R. Dssouli editors), North Holland, June 1999.

[7] 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.

[8] ETSI TC MTS. *TTCN-3 - Core Language*. European Norm (EN) 00063-1 (provisional)[2], 2000.

[9] ETSI TC MTS. *TTCN-3 - Tabular Presentation Format*. EN00063-2 (provisional)[2], 2000.

[10] ETSI TC MTS. *TTCN-3 - MSC Presentation Format*. EN00063-3 (provisional)[2], 2000.

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

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

[13] ITU-T SG 10. *Message Sequence Chart (MSC)*. Rec. Z.120, Geneva 2000.

[2] NOTE: The EN-00063 numbers are only provisional ETSI Work Item numbers. The actual EN numbers will not be the same.