HyperMSCs with Connectors for Advanced Visual System Modelling and Testing

Jens Grabowski[1], Peter Graubmann[2], and Ekkart Rudolph[3]

[1] Universität zu Lübeck, Institut für Telematik
Ratzeburger Allee, D-23538 Lübeck, Germany
grabowsk@itm.mu-luebeck.de
[2] Siemens AG, Corporate Technology, Software and Engineering
Otto-Hahn-Ring 6, D-81739 München, Germany
peter.graubmann@mchp.siemens.de
[3] Technische Universität München, Institut für Informatik
Arcisstraße 21, D-80290 München, Germany
rudolphe@informatik.tu-muenchen.de

Abstract. Experiences with the use of the MSC language for complex system specifications have shown that certain extensions are necessary in order to arrive at sufficiently transparent and manageable descriptions. Extended HMSCs, where MSC reference symbols may either be presented by hypertext-like descriptions or, in an expanded form, as detailed MSCs, appear to be especially suitable for a compact and transparent MSC representation. For an effective usage of such advanced MSC constructs, a corresponding tool support seems to be mandatory where interactively the event structures of special paths can explicitly be expanded while others remain hidden as MSC references that contain solely textual descriptions. The name '*HyperMSCs*' is proposed for such extended HMSCs. Beyond that, the communication between MSC references, operator expressions or HMSCs demands a generalisation of the gate concept. For that purpose, the introduction of MSC *connectors* denoting logical connections is suggested. MSC connectors may be expanded similar to MSC references. HyperMSCs enhanced by MSC connectors also provide a means for a selected visualisation of large MSCs in an interactive manner where, depending on the current selection, some parts are exhibited in full detail whereas other parts are presented in an abbreviated form. The same concepts may be applied for system modelling based on stepwise refinement starting with HyperMSCs, decomposed instances and MSC connector communication and for system testing.

1 Introduction

While Message Sequence Charts (MSCs) without any doubt are amongst the most popular and successful description techniques now, their real potential has not yet been exploited. Although the MSC language [8, 15, 22] contains very powerful composition mechanisms and other structural concepts, the language is still used essentially to define a set of sample behaviours. However, with the

increasing popularity of the MSC language, a more comprehensive application is demanded by some user communities. Recently, MSCs have been applied for a graphical presentation format for TTCN-3 test cases [16, 17, 19] and appears more intuitive than the tabular presentation format of TTCN-3. Experience with a comprehensive MSC based specification has shown that the MSC language needs certain extensions to facilitate reading and understanding of MSC diagrams, and to support and ease the handling of MSC documents. These extensions are strongly related to a corresponding advanced tool support. Hypertext-like mechanisms have been suggested for an appropriate handling of large MSC documents [6, 16, 17]. Because of this hypertext analogy, the term 'HyperMSC' has been introduced.

More generally, HyperMSCs can be viewed as a means for a selective detailed visualisation of certain parts of MSC, and consequently a hiding of other parts. With suitable tool support, such a selective visualisation may be quite flexible:

- MSC references may be folded and unfolded presenting their contents in a highly interactive manner; or
- decomposed instances can be used as a special kind of MSC reference to hide the message interaction description between selected instances if these messages actually are not in the focus of interest.

In addition to MSC references interpreted in a hypertext-like manner, the introduction of the *MSC connector concept* into the MSC language may be valuable for the same purpose. For complex MSC operator expressions MSC connectors have been proposed as a generalisation of the gate concept in [12, 13]. They may be used quite generally for the hiding of the detailed description of the message flow between MSC references or decomposed instances. Furthermore, MSC connectors may be used to predefine communication patterns that could be re-used in different places thus providing for the compositionality of MSC specifications.

In the next section, a brief description is given of the MSC presentation format developed for TTCN-3 test cases at ETSI. This provides a motivation for the concepts of HyperMSCs and MSC connectors. Within Sect. 3, the concept of HyperMSCs is elaborated and demonstrated using an example of the CCBS service specification. Sect. 4 is dedicated to the introduction and elaboration of the MSC connector concept and in particular to the inclusion of MSC connectors into the HyperMSC concept. In Sect. 5, a summary and outlook is provided.

2 Motivation: A HyperMSC-Based Presentation Format for TTCN-3 Test Suites

An MSC-based presentation format for TTCN-3 [19] has been developed as part of the ETSI Specialist Task Force on "Specification of a Message Sequence Chart/UML format, including validation for TTCN-3" – [16, 17, and ETSI STF 156]. Experiments with different variants of MSC representations

have shown that extensions of the MSC language are required in order to obtain a sufficiently transparent and readable MSC test description.

The most obvious and straightforward way to represent TTCN test cases by MSC diagrams would be to use inline operator expressions for alternatives, iterations, etc. Practice has shown that apart from simple cases, such a naive translation does not lead to diagrams that are easy to read and understand. In particular, inline operator expressions obscure the message flow for important cases: paths leading to a PASS verdict (PASS cases) are mixed with INCONC cases (paths leading to an INCONCLUSVE verdict) and FAIL cases (paths leading to a FAIL verdict).

The obvious drawback of this representation is the fact that PASS, INCONC, and FAIL cases are all presented in the same way. Certainly, it would be advantageous to have a means to emphasise the PASS cases in the test representation. The MSC language already contains several structuring mechanisms, and for this special purpose, the MSC reference mechanism seems to be suited. To highlight the PASS cases, all other cases may be represented as MSC references. Yet this notation has an immediate drawback: without explicitly looking into the definitions of the referenced MSCs, there is no immediate information about the hidden cases. This means that, the representation of the complete test case is not very intuitive except for the presented message exchange.

A much more satisfactory representation is obtained if a comment together with the test verdict is included in the MSC reference symbols instead of reference names. However, in the case of many alternatives the resulting representation is still not sufficiently transparent. As a general rule, inline expressions should be used only in a very limited manner and remain restricted to one or two alternatives or loops. In more complex situations, HMSCs are much more transparent since they abstract from details and focus on the compositional structure. However, if the inline representation is translated into an HMSC, we are faced with the problem to represent the expanded parts because (according to the standard) an HMSC contains only non-expanded MSC reference symbols. In order to overcome this deficiency, HyperMSCs are introduced which admit expanded MSC references within the HMSC formalism [10]. By shifting the branching points to the borderline, the PASS case is shown in expanded form in a coherent manner whereas the INCONC and FAIL cases are indicated as non-expanded MSC references. With advanced tool support the roles of the PASS and the INCONC cases may be changed interactively, and the HyperMSC document becomes a powerful tool for reading, presenting and analysing test cases.

A significant difference between TTCN and MSC shows itself in the behaviour description of several test components. TTCN-3 defines the behaviour of components of the same test case in separate functions. In corresponding MSCs this partitioning results in test component specifications that are merged by join operations. For an appropriate specification of such a join operation, an MSC connector has been proposed which allows description of the exchange of co-ordination messages among test components on a suitable level of abstraction.

HyperMSCs and MSC connectors are general concepts that allow an intuitive presentation of comprehensive communication patterns and behaviour. Their usage will not be restricted to the testing community only and thus, both concepts, we propose, should find their way into the MSC language.

3 From HMSCs to HyperMSCs

In the case of a great number of alternatives, the inline branching constructs for MSCs are not very transparent. In particular, they obscure the presentation of the complete message flow. In UML Sequence Diagrams, there is a tendency to indicate branching and iterations at the borderline of the diagram either using special graphical constructs, such as **for** loops, or even using a program-like form. In this way, the diagrams may be focussing on the representation of the pure message flow. In practice, however, such a notation again soon becomes quite intricate and clumsy, particularly for nested alternatives or loops. In the following, we re-formulate High Level MSCs (HMSCs) in such a way that they serve for a similar purpose. The obtained notation turns out to be very intuitive and simple, even in more complicated situations.

HMSCs describe the composition of MSCs as a graph with MSC references and conditions as nodes [10, 15]. This way, they abstract from instances and messages which are not shown in the diagram. Each MSC reference has a name. Its meaning is given by a corresponding MSC with the same name defined in another place in the same MSC document. HMSC diagrams with many references that refer to many fairly small MSC definitions, soon become again quite complex and difficult to maintain and handle.

In order to arrive at user-friendlier handling, HMSCs may be re-interpreted in a way that has an analogy in hypertext-like specifications. MSC references may be shown optionally also in an expanded manner within the HMSC and non-expanded MSC references may contain hypertext-like descriptions instead of pure reference names [6, 16]. This implies a real extension of the MSC standard, but essentially concerns the handling and the graphical layout only: it has no effect on the semantics. Thereby, we assume a corresponding tool support where the MSC references can be interactively expanded within the HMSC in which they are contained or are possibly in a separate window (for example double clicking into an MSC reference symbol to fold or unfold it, etc.). Obviously, in this context tools should have appropriate mechanisms and adequate means to define MSC references inline.

Drawing expanded MSC references in HMSC presentations has been already suggested in [10], but without the additional dynamic mechanisms of the HyperMSCs, this only inflates the diagrams and makes them even harder to understand. In addition, gates were introduced in [10] which proves to be not completely appropriate. Our proposed solution for this problem is the connector concept which will be presented in Sect. 4.

In Fig. 1, the concept of HyperMSCs is indicated schematically. Fig. 1(a) shows an HMSC with connection points between the MSC references $R1$ and

$R2/R4$ and between the MSC references $R2$ and $R3/R5$. In Fig. 1(b) the MSC references $R1$, $R2$, and $R3$ appear in expanded form within the HMSC diagram.

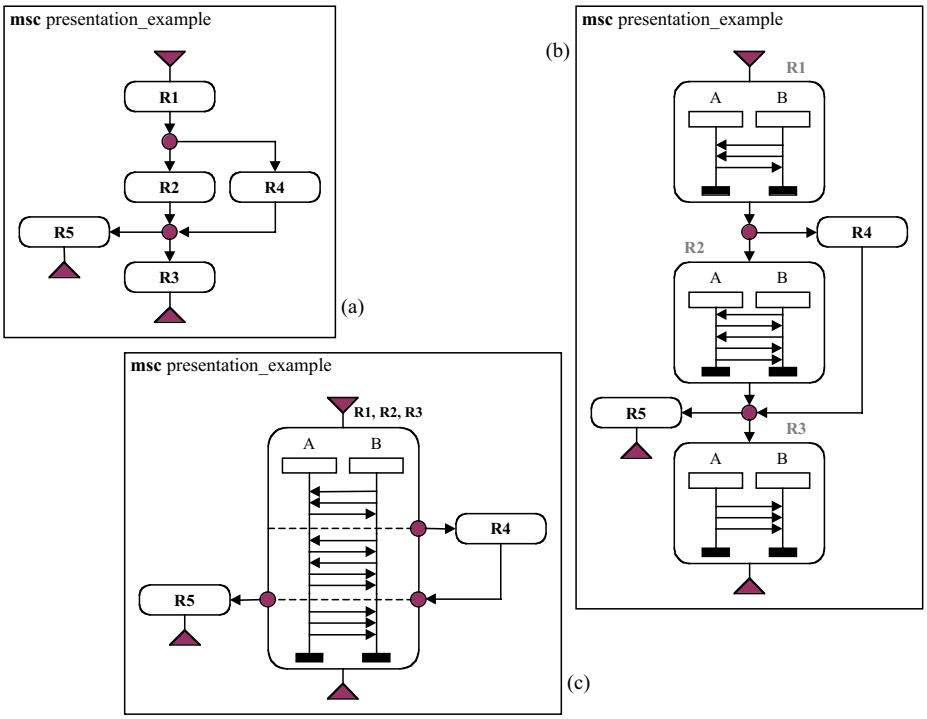

Fig. 1. Various representations of one HyperMSC
(a) as HMSC,
(b) with expanded MSC references,
(c) with expanded MSC references as coherent "path of interest".

The HyperMSC that results if we only admit expanded MSC references still has the drawback that the main event flow usually is split into many separate parts. In the case of many alternatives describing exceptional or erroneous behaviour, this splitting is very disturbing since it is not possible to show the main flow in a coherent manner in expanded form. 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 HyperMSC itself. Therefore, a further extension of HMSCs has been suggested which somehow may be viewed also as a unification of High Level MSCs and plain MSCs. Several expanded MSC references, which are interrupted by branching points, may be combined to one coherent MSC reference. As a consequence, the branching (or connection) points have to be shifted to the borderline of the resulting MSC reference. In Fig. 1(c), the MSC references $R1$, $R2$ and $R3$ are combined to one coherent message flow while the connector points are shifted to the borderline.

Such a coherent representation is possible also in the case of cyclic or compound HMSCs, and thus may cope with much more complex situations than just few alternatives. For nested alternatives the hierarchical structuring of HMSCs is a major advantage. HMSCs are hierarchical in the sense that a reference in an HMSC may again refer to an HMSC.

Particular attention has to be paid to keep the different appearances of the MSC references identified. Within HyperMSCs, MSC references can be displayed in several ways:

1. as MSC reference symbol with the proper name of the reference inscribed (see Fig. 1(a));
2. as MSC reference symbol with an explanatory text, the inscription (see Fig. 3(b));
3. as expansion of the MSC reference (see Fig. 1(b) and (c)).

One may imagine other representations as well: program code fragments representing the behaviour of the MSC, TTCN-3 test definitions, etc. However, it is important, that in all but the case (1), the MSC reference name is attached additionally to the presentation, to unambiguously identify expanded elements. There are several possibilities to attach the MSC reference name (see Fig. 1(b) and (c) where the names of the expanded elements are placed on top of the expansion, or Fig. 6 where the reference to the expanded MSC references is to be found after the dashed line which separates the expansion from the previous one).

It has to be pointed out, that the expansion of MSC references within the HyperMSC concept is not purely a tool issue, but means a real graphical extension of the MSC language because MSC reference symbols containing detailed MSCs are not allowed in the MSC standard language. This extension is in our view most important since the dynamics behind the HyperMSCs (the expanding and folding of MSC references, etc.) is an essential means to increase the understandability of MSC diagrams, in particular if self-contained parts of a system have to be described completely.

The concept of HyperMSCs can also be carried over to MSC operator expressions. For that, we want to recall the relation between MSC operator and inline expressions. As can be seen in Fig. 2, inline expressions just describe unfolded MSC reference operator expressions. For MSC operator expressions the unfolding is well established (except for the operator **seq).**

3.1 Integration of HyperMSC into the MSC Language

HyperMSCs provide the concepts for an MSC presentation that allow an interactive folding or unfolding of parts of the MSC diagrams. Of course, the full benefits of these concepts can be only realised with adequate tool support. In general, it has to be decided manually what is to be folded and what is to be displayed in full detail. However, sophisticated tools may analyse HMSCs and MSC diagrams and display them in an optimal form (according to built-in strategies

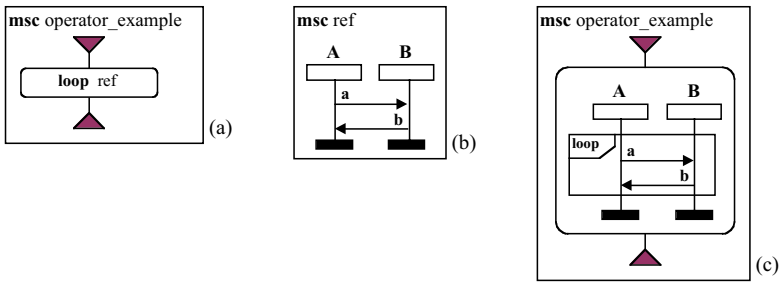

Fig. 2. Expansion of an MSC operator expression
(a) with its MSC reference (b) by means of an inline expression which results in (c).

or user predefinitions). For tool interaction and the exchange of diagrams, the MSC/PR language plays an important role, thus, the HyperMSC concepts have to be reflected in the textual presentation form, too. We propose the integration of the HyperMSC concepts into the MSC/PR language by augmenting of the MSC/PR with XML tags.

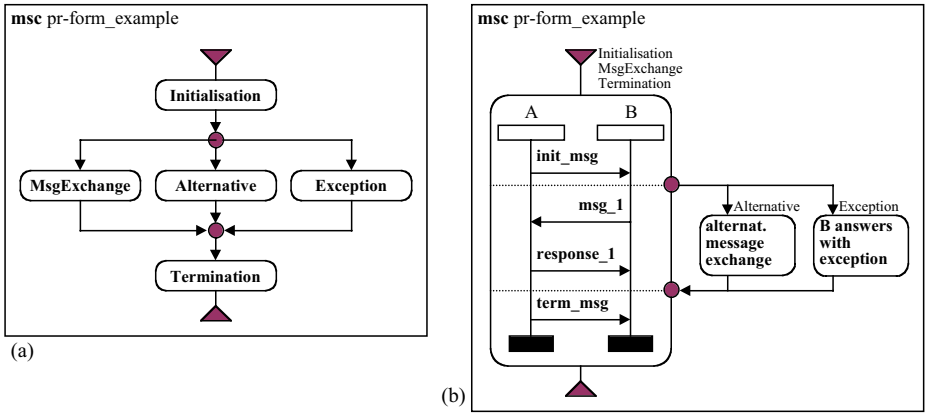

Fig. 3. HyperMSC presentation of an MSC
(a) completely folded; (b) partially expanded

A small example may provide a first idea of what an HyperMSC consistent extension of the MSC/PR may look like. Fig. 3 shows two variants of the graphical representation of a HyperMSC. Fig. 3(a) is the classical HMSC presentation, Fig. 3(b) is one of its graphical presentation variants with three expanded MSC references and two folded ones, showing the explanatory 'inscription' in lieu of the reference name.

The corresponding PR form is presented in Fig. 4. In addition to the basically unchanged structural description of the HyperMSC, it contains the XML tag `hyperref` which controls the GR form presentation of the MSC references. The attributes of the hyperref tag are `representation` (indicating whether the MSC reference is to be presented folded or unfolded, showing its proper name or with an explanatory description attached, etc.), `inscription` (containing the text of

```
msc pr-form_example;
expr L1;
L1: <hyperref
      representation=folded&name
      inscription=no-inscription
      expansion='C:\>Initialisation.msc'>
    (Initialisation) <\hyperref> seq (L2);
L2: connect seq (L3 alt L4 alt L5);
L3: <hyperref
      representation=folded&name
      inscription=' expected message exchange'
      expansion='C:\>MsgExchange.msc'>
    (MsgExchange) <\hyperref> seq (L6);
L4: <hyperref
      representation=folded&name
      inscription='alternat. message exchange'
      expansion='C:\>Alternative.msc'>
    (Alternative) <\hyperref> seq (L6);
L5: <hyperref
      representation=folded&name
      inscription=' B answers with exception'
      expansion='C:\>Exception.msc'>
    (Exception) <\hyperref> seq (L6);
L6: connect seq (L7);
L7: <hyperref
      representation=folded&name
      inscription=no-inscription
      expansion='C:\>Termination.msc'>
    (Termination) <\hyperref> seq (L8);
L8: end;
endmsc;
```

(a)

```
msc pr-form_example;
expr L1;
L1:  <hyperref
        representation=expanded&init
        inscription=no-inscription
        expansion='C:\>Initialisation.msc'>
     (Initialisation) <\hyperref> seq (L2);
L2:  connect seq (L3 alt L4 alt L5);
L3:  <hyperref
        representation=expanded&continued
        inscription=' expected message exchange'
        expansion='C:\>MsgExchange.msc'>
     (MsgExchange) <\hyperref> seq (L6);
L4:  <hyperref
        representation=folded&inscription
        inscription='alternat. message exchange'
        expansion='C:\>Alternative.msc'>
     (Alternative) <\hyperref> seq (L6);
L5:  <hyperref
        representation=folded&inscription
        inscription=' B answers with exception'
        expansion='C:\>Exception.msc'>
     (Exception) <\hyperref> seq (L6);
L6:  connect seq (L7);
L7:  <hyperref
        representation=expanded&continued
        inscription=no-inscription
        expansion='C:\>Termination.msc'>
     (Termination) <\hyperref> seq (L8);
L8:  end;
endmsc;
```

(b)

Fig. 4. Textual representation (PR form) of the HyperMSC 'pr-form_example' in Fig. 3
(a) textual representation of the completely folded HTML representation;
(b) textual representation of the partial unfolded variant of Fig. 3(b).

the explanatory description), `expansion` (referencing the location of the MSC reference expansion), etc.

The instrumentation may not be restricted to special MSC language constructs. It should be possible to identify arbitrary parts of MSC and HMSC diagrams that can be folded to HyperMSC references and when required can be re-expanded. The HyperMSC mechanism adds the possibility to structure information on the presentation level to the MSC language.

3.2 Usage of HyperMSC

HyperMSC is a means to emphasise selected behaviour and to abstract behaviour alternatives that are currently less relevant. Hypertext-like inscriptions within MSC reference symbols provide a natural interface to different MSC user communities such as system designers, system developers and test engineers. For system designers the HyperMSC concept provides a tool that allows specifying system behaviour in the form of purely textual descriptions that later on are refined into the form of concrete MSCs. For other user communities, HyperMSCs may be used to guide the understanding and manipulation of MSCs by emphasising behaviour relevant for the momentary analysis and investigation and providing additional support in the form of explanatory descriptions placed into MSC reference symbols instead of the MSC reference name.

As has been outlined in Sect. 3, the HyperMSC concept was stimulated by the development of a graphical test format. However, HyperMSCs may in fact have a

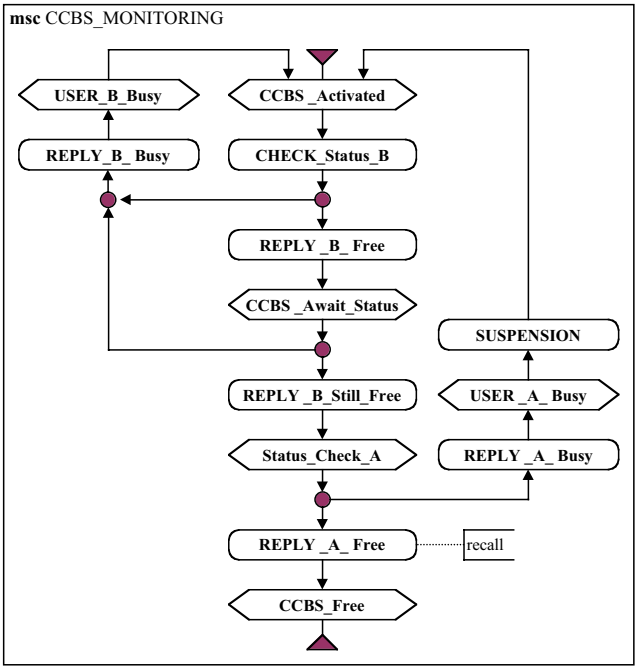

Fig. 5. CCBS example: Monitoring

much larger area of application. In Fig. 5 and Fig. 6, the usage of the HyperMSC concept is demonstrated by means of an extract of the CCBS (Completion of Calls to Busy Subscriber) service specification (ISDN) [15]. The HMSC in Fig. 5 which describes only a fairly small and simplified part of the CCBS service contains already quite a considerable number of MSC references. Since in standard MSC these MSC references are defined by separated MSC diagrams within the MSC document, so that the representation consists of many small pieces rather than providing a coherent view. As a consequence, such a specification would fail the main purpose of the MSC language. Obviously, a much more satisfactory and convincing representation is provided by means of the corresponding HyperMSC in Fig. 6, in which the main path is shown in an expanded and coherent manner. The side cases contained in the non-expanded MSC references may be expanded individually in a hypertext-like manner. The chosen CCBS example demonstrates that within HyperMSC alternatives as well as cyclic behaviour may be represented in a convincing and transparent form, even in more complex cases.

4 MSC Connectors

For a long time the MSC language only offered the communication pattern of basic message exchange: a non-blocking asynchronous dispatch of information from a sender to a receiver. The means to express a blocking synchronous communication has only recently been introduced (see also [6, 14]). However, there

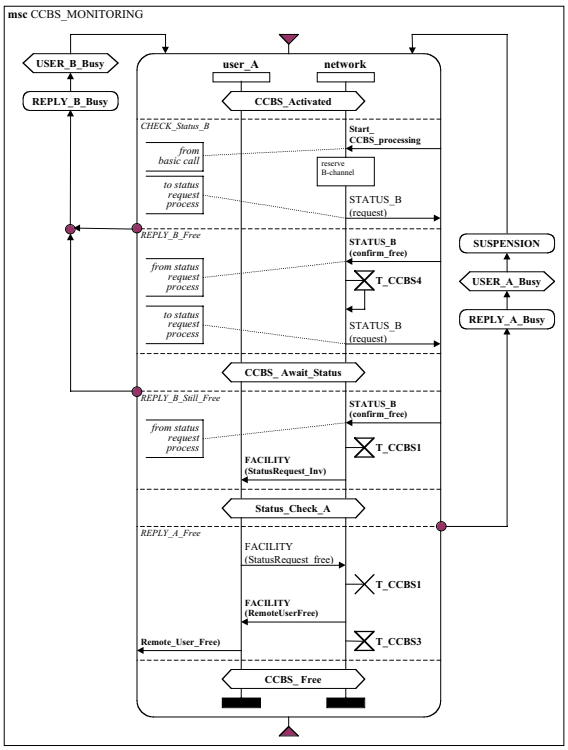

Fig. 6. CCBS example: Monitoring partially expanded

are not yet language constructs that provide for further abstractions of the information interchange – in contrast to the well known MSC references and operator expressions which serve as behaviour abstractions. The need for communication abstractions, however, is obvious. There are three main reasons for their introduction:

1. *MSC language completeness:* Behaviour abstractions indeed need the accompanying abstractions of the information exchange, which in this case is just the well-defined folding of the messages coming in and going out of MSC operator expressions (find more details below).
2. *Practicability of MSCs for describing complex systems:* To exploit MSCs as a description technique for practical system design (beyond the mere specification of a few exemplary behaviour traces), abstraction is still the key issue in handling the system complexity. Compositionality and re-use are in the centre of modern description techniques, because today's system development is component oriented and deals with system families and product lines. In this context, the communication between the parts of a system (components, objects, etc.) will follow interaction patterns which are expressed in the component oriented world as software connectors [11]. They are defined once then implemented and then re-used over and over again (see system families where interaction patterns are essential). MSC connectors mirror these facts and allow separate definition of recurring interaction patterns

and joint application with the other MSC abstractions. The same issue is stressed in Sect. 2 where the adequate abstractions for the test descriptions are presented.

3. *Satisfactory HyperMSC presentation techniques:* MSC connectors appear as a by-product of the HyperMSC concept. Folding and unfolding of parts of an MSC diagram is essential to present an MSC in a form where its interesting parts are shown in full detail but all others are folded away. To provide a correct and comprehensive MSC representation in spite of the folding, a well-defined abstraction mechanism has to be in place (see Fig. 13). The consequent elaboration of the HyperMSC concept leads to mechanisms that allow either folding or unfolding (parts of) instances to MSC references and to subsume groups of messages into MSC connectors.

In this paper, we focus our discussion on the issue (1) and the issue (3) above, where MSC connectors can be employed in a natural, somewhat simplified and default-based manner. Issue (2), which shows the great potential of the MSC connector concept, is bound to utilise more general mechanisms to reach its full effectiveness. This issue will be detailed in a further publication (see also [7]).

The introduction of MSC connectors appears to be inevitable in order to clearly define the message communication between general MSC operator expressions [10, 12, 13]. The current MSC standard is arguably not precise enough in this point by using (or perhaps abusing) normal messages as sort of MSC connectors. Actually the standard MSC language uses message gates to define the connection points for messages with respect to the interior and exterior of MSC references and inline expressions. The new MSC connector concept is supposed to elegantly subsume and generalise the gate construct.

Within Fig. 7(a), an example is provided showing two inline loop expressions connected by one message which in fact has the meaning of three connecting messages (as indicated in Fig. 7(b)). Yet, the message m in Fig. 7(a) between the two loop expressions in fact denotes an MSC connector and thus should be graphically distinguished (see Fig. 7(c)).

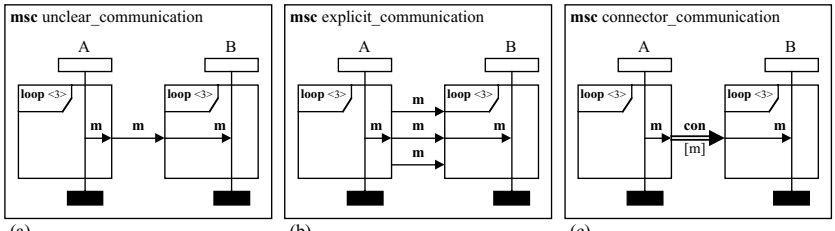

Fig. 7. Message communication between loop expressions
(a) rather unclear description according to the current MSC standard;
(b) explicit description; (c) description using an MSC connector.

From this example, we see that a message specification in an MSC defines an actual occurrence of a message (the sending and receiving event) whereas an MSC connector alone only denotes the *possibility* of message occurrences.

Actually, the MSC connector needs a definition that determines what kind of communication pattern it represents. In the above example, the connector *con* may be defined as accepting an arbitrary number of messages m (in Sect. 4.1, the explicit definition of MSC connectors will be briefly discussed).

MSC connectors between MSC operators and references bundle the messages that are crossing the environment of the references (see Fig. 8). Within an MSC reference, the *connector pointer* indicates to which connector a message is sent (`<message_name>` → `<connector name>`) or from which connector it is received (`<connector_name>` → `<message_name>`). The connector itself is presented graphically as a double lined arrow (possibly with two arrow heads, if the connector represents a bi-directional communication). As well as its name the connector is associated with the message list which indicates the names of the messages that are admissible to the connector (see Fig. 8(a) with the message list $[a, b]$ attached to the connector *con*).

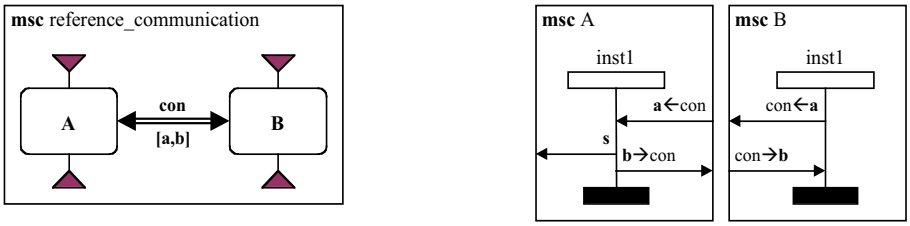

Fig. 8. Communication between MSC references via an MSC connector

Experiments with MSC connectors in large applications will determine how defaults for connector pointers, message lists, etc., will be optimally chosen. There could be rules such as "For an MSC reference, all messages from and to the environment that are listed in the message list of an attached connector are passed through this connector".

4.1 MSC Connector Specifications

MSC connectors do not only bundle a bunch of messages between MSC references, they denote a particular behaviour. Depending on the application, various communication mechanisms may be assigned to MSC connectors. Therefore it becomes necessary to explicitly specify the behaviour of each MSC connector prior to its usage or to use pre-defined default connector types (as we did in the examples given here, assuming the consistency as given[1]). In the following, we discuss a few MSC connector types that may be useful in general and we briefly indicate which mechanisms can be used to define the connector behaviour:

The unrestricted connector: In its most general form an MSC connector transmits messages in arbitrary order if they are named in its message list. Whether these messages cross each other is not determined. Therefore the order relation

[1] We also suppressed the necessary typing of the MSC connectors in the examples to clearly present the concepts without going into mere technicalities.

between sending and consumption of messages is exclusively specified by the respective event definitions given by the connected components.

The FIFO connector: The FIFO connector is probably the connector type which describes the most common situation. It is capable of accepting messages in an arbitrary order, as long as they are named in its message list. However, in contrast to the unrestricted connector, messages from the connected components come out of the connector in the same order in which they are put onto it: messages must not overtake (FIFO property). A FIFO connector can transport a message only if it has been put onto the MSC connector first. As a consequence, alternative or loop operands which are initiated by an input message coming out of an MSC connector are enabled only if this message has been put onto the connector first – and (of course) if any guards that are present evaluate to true.

Similarly, a LIFO connector may be defined which conveys messages to a connected component in inverse input order.

The unrestricted message list: Message lists determine the names of the messages that are admitted into an MSC connector (an example of the usage of message lists is to be found in Fig. 8). It proves convenient to define an unrestricted message list, which allows all messages to be sent and retrieved from a connector. This unrestricted message list is denoted as $[*]$, but may be omitted.

Regular expressions for the description of more complex communication patterns: Because MSC connectors are assumed to support the specification of systems or components, a certain behaviour is associated with them. Both connectors described above represent connectors with a very unrestricted behaviour. They can be conveniently applied in many cases. In other cases, however, where systems are composed out of existing or separately defined components, it is appropriate to associate connection semantics with connectors and to use them with a more restricted behaviour. To allow for such a behaviour definition, regular expressions over message names may be assigned to an MSC connector (see the example in Fig. 9) or separately to the connector endpoints (this allows an explicit specification of crossing messages, though some consistency requirements apply). The regular expression is attached to the connector and enclosed in guillemet brackets (for example $\ll\{ab\}*\gg$). An omitted regular expression means the behaviour of the connector unrestricted.

Any behaviour of the connected components that does not match with the connector definition is excluded (see Fig. 9 for an informal explanation).

MSC representation of connector behaviour: It is easy to imagine, that the most convenient way to define the behaviour of an MSC connector is by providing it as another MSC. This also will provide an appropriate framework to present the formal semantics of the connector concept based upon the synchronisation of event traces. However, a few technicalities are involved in presenting this coherently, therefore a detailed presentation is postponed to a forthcoming paper (meanwhile, see [7] for details).

It may also be of interest to point out the differences of MSC connectors and SDL channels. In SDL, channels are part of a static architectural specification: they connect outputs and input queues of processes. With respect to the

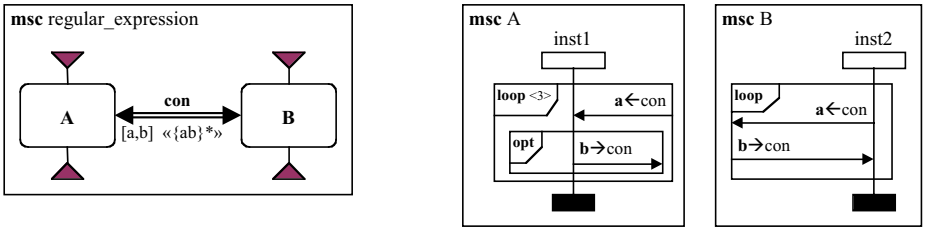

Fig. 9. Communication of MSC references via an MSC connector with explicit behaviour specification

The MSC reference A is able to produce the message sequences $\ll\{\{a|ab\}^3\}\gg$, B is able to produce the message sequences $\ll\{ab\}*\gg$. Due to the connector definition, the composed MSC produces the message sequence $\ll\{ab\}^3\gg$.

dynamic specification, an output corresponds to the send event and the input queue to the reception event of messages. In MSC, however, the message events normally are interpreted as sending and processing events. To specify the reception event, the inclusion of additional instances representing the behaviour of the input queue would be necessary (which could be done transparently with an appropriately defined MSC connector). Beyond that, the MSC connector construct is introduced on a highly abstract level denoting a purely logical communication construct whereas in SDL, a channel is predominantly static and usually closer to the description of a realisation mechanism.

4.2 Combining MSC Connectors with the HyperMSC Concept

The inclusion of MSC connectors into the MSC language also leads to a generalisation of the HyperMSC concept described in Sect. 3. Where simple MSCs without branching or loops define MSC references that are joined by a connector, this generalisation appears to be quite straightforward: the MSC references may be expanded as usual. In addition, the MSC connector may be expanded thus exhibiting the detailed message communication between the MSC references. An example is provided in Fig. 10. Figure 10(a) shows the HMSC *reference_communication* of Fig. 8 with expanded MSC references. In Fig. 10(b) also the connector *con* is expanded, thus connecting the messages a and b of MSC reference A and MSC reference B to one coherent message flow.

The situation is more complicated if the connected MSC references are defined by means of MSCs containing alternatives as in Fig. 11. The MSC references A and B in MSC *alternative_communication* are defined in form of HMSC A and HMSC B which contain as alternative branches the MSC references $A1/A2$ and the MSC references $B1/B2$, respectively. Obviously, MSC $A1$ only can be matched with MSC $B1$ and MSC $A2$ only with MSC $B2$. As a rule, it should therefore be only allowed to expand corresponding alternatives which fit together with respect to the connector communication. For example, in Fig. 12(a) the corresponding alternatives MSC reference A1 and MSC reference $B1$ are presented in expanded form. Fig. 12(b) provides a representation showing MSC reference $A1$ and $B1$ together with the connector *con* in completely expanded form.

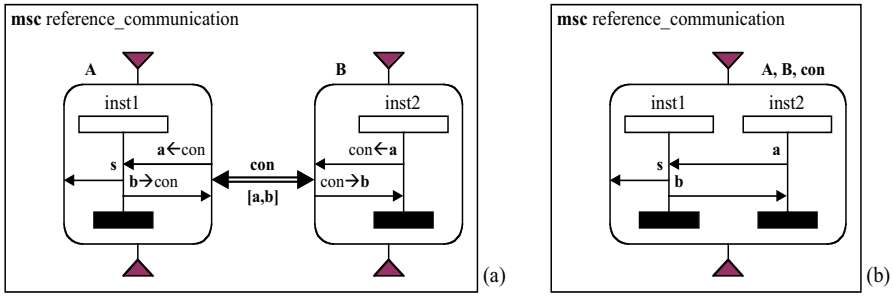

Fig. 10. Unfolding of MSC references and MSC connectors (see Fig. 8)
The HyperMSC presentation with folded MSC references A and B): (a) Unfolding of the MSC references and (b) jointly unfolding of the MSC references and the MSC connector.

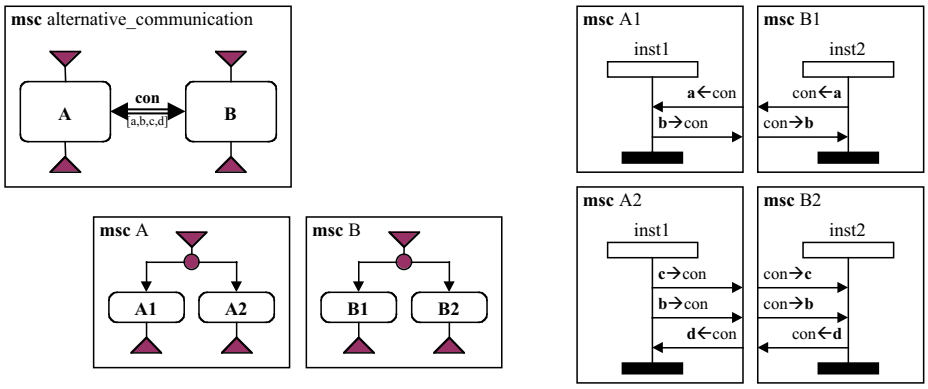

Fig. 11. Definition of the MSC alternative_communication

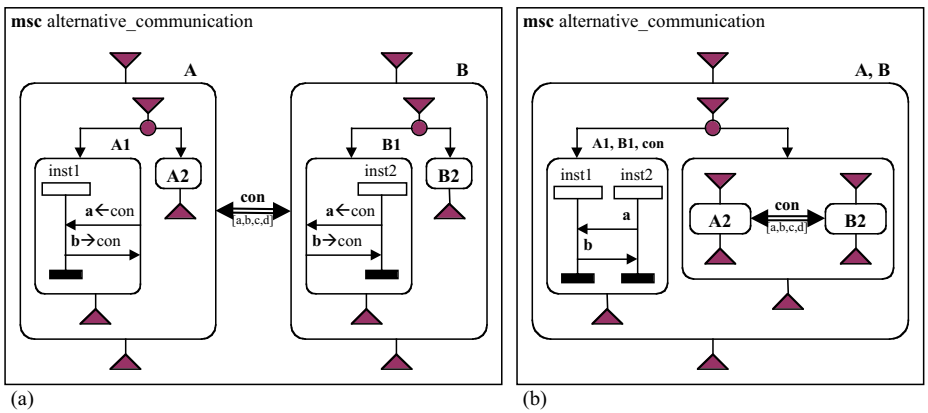

Fig. 12. Two variant HyperMSC presentations of the MSC defined in Fig. 11

The previous examples demonstrate how HyperMSC diagrams present themselves if MSC references and MSC connectors are folded and unfolded. There is a large number of possibilities to select, group, and present the information contained in the MSC diagrams in a way that is optimal and shows exactly what is considered relevant for a particular analysis step, a structured walk-through, etc. The folding and unfolding mechanisms of HyperMSCs therefore provides a means to cope with the complexity of a system design and its large amount of information. However, the examples given so far started with already defined MSC references and MSC connectors. This is a natural way, since system design and modelling just produces all these abstractions that are consequently represented by MSC references and connectors. Yet, there is a further potential in the flexibility of the presentation means proffered by HyperMSCs: the ad-hoc folding/unfolding of diagrams and their parts. Such presentation does not impact on the structure of the model, but focuses on the display of those parts of the MSC diagram that are relevant at the very moment of viewing it. Therefore ad-hoc folding may be used to highlight a statement or an idea during a discussion, to concentrate on a particular trait during analysis, etc. Altogether, folding supports the designers' actual work with the diagrams. However, it is essential that the context of a selected diagram part does not completely vanish, but remains visible even if only in a condensed form.

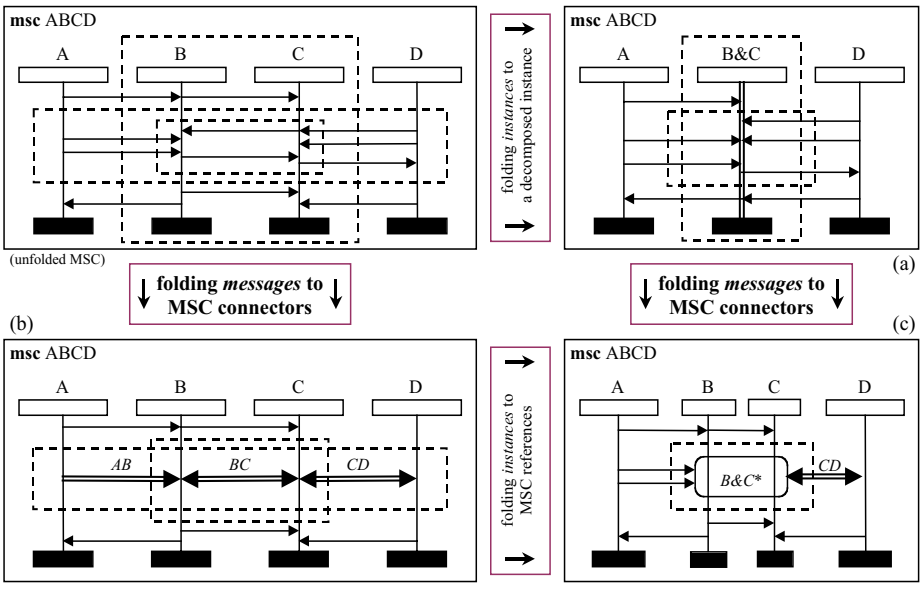

Fig. 13. Ad-hoc folding of a HyperMSC

(a) Folding of instances; (b) folding of a set of messages; (c) folding of a selected part of an MSC.

Figure 13 tries to give an impression about what ad-hoc folding (and unfolding) may look like. Folding away instances (Fig. 13(a)) produces decomposed instances (the instance $B\&C$) which are, tentatively, drawn with a double line, just in analogy to the double lined arrow of the connector symbol. Folding away groups of messages (Fig. 13(b)) produces ad-hoc defined connectors which take over the characteristics of the messages that are subsumed within it (cf. the one-directional connector AB, containing two messages directed from instance A to instance B). The most interesting folding operation is to fold away a part of the diagram that covers both, instances and messages (see Fig. 13(c)). Interactions between the part of the instances that are going to be hidden result in MSC references (see the reference $B\&C*$), messages can be grouped into connectors (as is done with the generation of the MSC connector CD), or may be displayed in the classical way as messages just entering the MSC reference. The wide variety of possibilities and the high flexibility to switch between different presentations allows to quickly come up with a presentation that is optimally adjusted to the actual demands.

5 Summary and Outlook

Within this paper, the HyperMSC concept which has been introduced recently [6, 16, 17] has been further elaborated and enhanced by MSC connectors representing communication patterns on a highly abstract level. HyperMSCs including MSC connectors are ideally suited to present MSCs on different levels of detail and, what is most important, to easily switch between different views. Assuming an advanced tool support, such enhanced HyperMSCs may be also used for a visual preparation of large MSCs which is highly interactive. By defining MSC references, decomposed instances and MSC connectors in an interactive manner, certain details of a large MSC may be hidden in order to exhibit the momentarily relevant parts. HyperMSCs have already been successfully applied to the specification of test cases based on TTCN-3 and Use Case modelling within UML. At present, the ETSI test format based on HyperMSCs with MSC connectors is under preparation within OMG for inclusion as a profile to UML. The MSC connector concept may be further generalised, eventually representing a complex message interface between system components.

References

1. F. Belina, D. Hogrefe, A. Sarma: *SDL with Applications from Protocol Specification.* Prentice Hall, 1991.
2. G. Booch, J. Rumbaugh, I. Jacobson: *The Unified Modelling Language User Guide.* Addison-Wesley, 1999, 3^{rd} edition.
3. D.F. D'Souza, A. C. Wills: *Objects, Components and Frameworks with UML. The Catalysis Approach.* Addison-Wesley, 1999.
4. A. Egyed, N. Metha, N. Medvidovi: *Software Connectors and Refinement in Family Architectures.* In: Proceedings of the 3^{rd} International Workshop on Software

Architectures for Product Families, Las Palmas de Gran Canaria, Spain, March 15-17, 2000.
5. J. Grabowski, A. Wiles, C. Willcock, D. Hogrefe. *On the Design of the new Testing Language TTCN-3*. In: Testing of Communicating Systems - Tools and Techniques (H. Ural, R.L. Probert, G. von Bochmann, editors), Kluwer Academic Publishers, August 2000.
6. P. Graubmann, E. Rudolph: *HyperMSCs and Sequence Diagrams for Use Case Modelling and Testing*. In: UML2000, 3^{rd} International Conference on The Unified Modeling Language (A. Evans, S. Kent, B. Selic, editors), 02-06 October, 2000, York, UK, Springer 2000.
7. P. Graubmann, R. Wasgint: *Methods for Interface Annotations and Component Selection*. SAG-WP2-0106-16, ESAPS internal report, 2001.
8. I. Krüger: *Distributed System Design with Message Sequence Charts*, PhD Thesis, Technische Universität München, 2000.
9. S. Loidl, E. Rudolph, U. Hinkel: *MSC'96 and Beyond-a Critical Look*. In SDL'97 Time for Testing-SDL, MSC and Trends, Proceedings of the 8^{th} SDL Forum in Evry, France (A. Cavalli and A. Sarma editors), North Holland, September 1997.
10. S. Mauw, M. A. Reniers: *High Level Message Sequence Charts*. In: SDL'97 - Time for Testing-SDL, MSC and Trends, Proceedings of the 8^{th} SDL Forum in Evry, France (A. Cavalli ,A. Sarma, editors), North Holland, September 1997.
11. N. Mehta, N. Medvidovic, S. Phadke: *Towards a Taxonomy of Software Connectors*. University of Southern California, Center of Software Engineering, Technical Report 99-529, 1999.
12. E. Rudolph: *Putting Extended MSC-2000 to Practice*, Contribution to the ITU-SG 10 Meeting, Geneva, November 1999.
13. E. Rudolph: *Advanced MSC- A Unifying Modeling Language for the Next Millennium*, Contribution to the ITU-SG 10 Meeting, Geneva, November 1999.
14. E. Rudolph, J. Grabowski, P. Graubmann*: Towards a Harmonization of UML-Sequence Diagrams and MSC*. In: SDL'99 - The Next Millennium, Proceedings of the 9^{th} SDL Forum in Montréal, Québec, Canada (R. Dssouli, G.V. Bochmann, Y. Lahav, editors), Elsevier Science B.V., Amsterdam, 1999.
15. E. Rudolph, J. Grabowski, P. Graubmann: *Tutorial on Message Sequence Charts (MSC-96)*. Forte/PSTV'96. Kaiserslautern, Germany, October 1996.
16. E. Rudolph, I. Schieferdecker, J. Grabowski: *Development of an Message Sequence Chart/ UML Test Format*. In: Proceedings of FBT'2000 - Formale Beschreibungstechniken für verteilte Systeme, Lübeck, Germany (J. Grabowski, S. Heymer, editors). Shaker-Verlag, Aachen, 2000.
17. E. Rudolph, I. Schieferdecker, J. Grabowski: *HyperMSC - A Graphical Representation of TTCN*. Proceedings of the 2^{nd} Workshop of the SDL Forum Society on SDL and MSC (SAM'2000), Grenoble, France, June, 26 - 28, 2000.
18. ETSI TC MTS: *TTCN-3 — Core Language*. European Norm EN00063-1 (provisional)[2], 2000.
19. ETSI TC MTS: *TTCN-3 — Graphical Presentation Format*. European Norm EN00063-3 (provisional), 2000.
20. ETSI TC MTS: *TTCN-3 — Tabular Presentation Format*. European Norm EN00063-2 (provisional), 2000.
21. ITU-T Rec. Z.120 (MSC-96): *Message Sequence Chart (MSC)*., Geneva, 1996.
22. ITU-T Rec. Z.120 (MSC-2000): *Message Sequence Chart (MSC)*., Geneva, 1999.

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