Message Sequence Chart: Composition Techniques versus OO-Techniques – 'Tema con Variazioni' –

Ekkart Rudolph[a], Peter Graubmann[a], and Jens Grabowski[b]

[a]Siemens AG, ZFE T SE, Otto-Hahn-Ring 6, 81739 München, Germany

[b]University of Berne, Institut for Informatics , Neubrückstr. 10, 3012 Berne, Switzerland

Structural concepts for Message Sequence Charts (MSCs), i.e., composition, types, inheritance, and virtuality, are applied to a telecom example provided by the public switching systems division of the Siemens AG. The example contains several variations of the peripheral parts of an initial MSC which may be combined independently. The independent combinations of the peripheral variations are described by means of several new composition operators and by using object-oriented techniques (OO-techniques), i.e. types, inheritance, and virtuality. A comparison of both techniques shows that composition operators may provide a compact, easy, but abstract description, whereas some OO-techniques allow a graphical, intuitive, but not compact specification. Typical OO-techniques like inheritance and virtuality seem to be less fruitful for the description of at least the provided example. A combination of composition operators and OO-techniques, e.g., a variant type concept employing the alternative composition operator, may combine the advantages of both techniques.

1. Introduction

The standardization work on MSC during the study period 1989 - 1992 has concentrated on the elaboration and standardization of basic concepts. Only few structural concepts, i.e., *coregion* and *submsc*, have been included in the MSC recommendation Z.120 [12]. For the present study period, the development of structural concepts for MSCs has become a central goal. In particular, composition and OO- techniques are in the focus of the on-going standardization activities [6]. This is not surprising. Without such concepts the usage of the MSC language would remain limited to the specification of few scenarios.

To develop a telecom service with MSCs means to handle a large number of MSCs. Until now, mostly conditions have been used to reduce their number and to indicate possible MSC combinations. Conditions, however, are not sufficient to keep the necessary sets of MSCs manageable. Syntactic means for a compact denotation of MSCs are missing in Z.120. Furthermore, reusability of (parts of) MSCs is not addressed in this document. And, indeed, MSCs are often identical save for minor variations or they have large parts in common (particularly, when during the development of telecom services country specific adaptations have to be taken into account). The development of concepts that deal with reusability and a more compact description of MSCs, however, pose several problems:

1. The new concepts have to provide the necessary support in a concise manner, i.e., only a few but powerful new concepts shall be defined.

2. Elegance and readability of MSCs must not deteriorate, i.e., all language constructs supporting the new concepts are required to be easily understood and handled in both textual or graphical representations.

3. An adequate semantics definition has to be provided that fits into the existing MSC semantics.

4. The level on which the new concepts are applied has to be determined. In ITU-T SG10 Q.9 so far the conception prevails that the required language constructs mainly apply to MSC documents. Thus, the introduction of the new concepts additionally requests the definition of the semantics of MSC documents.

This paper focusses on the first two points. A telecom example, provided by the public switching systems division of the Siemens AG, is used to demonstrate and analyze the proposed concepts. The example consists of an initial MSC which only describes behaviour abstraction and of further MSCs which describe several variations of the peripheral parts of the initial MSC. There is the necessity to describe the independent combination of the peripheral variations in a comfortable and compact manner. We applied both composition and OO-techniques in order to meet the user requirements and compared both modelling techniques.

The paper is organized in the following manner: The fundamental concepts of this case study are described in Section 2. In Section 3 composition techniques and OO-techniques are applied to the telecom example. The results of this application are discussed in Section 4. Section 5 presents summary and outlook.

2. Foundations

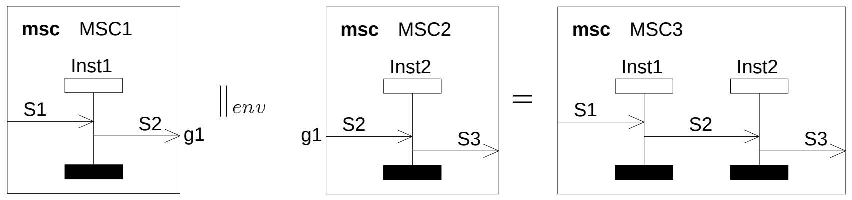
This section introduces several composition operators and OO-principles for MSCs in an informal manner. Only their basic ideas are sketched here, since an intuitive understanding is sufficient for our analysis.

2.1. Composition techniques

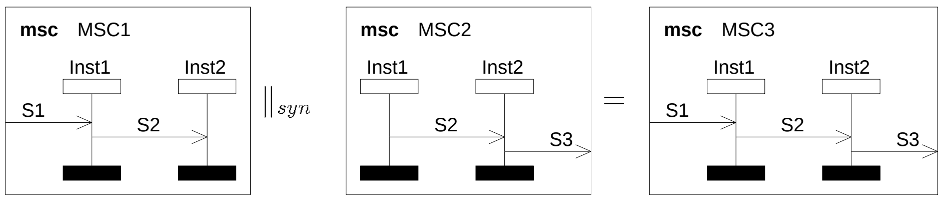
Three composition operators for MSCs are suggested: Environmental merge, synchronisation merge, and synchronisation condition merge. Each operator maps two MSCs into a new MSC.

The environmental merge operator ($\|_{env}$) identifies every message sent to or received from a gate in the environment of the first MSC (MSC1) with the message received from, respectively sent to, the equally named gate in the environment of the second MSC (MSC2). Figure 1 (a) presents an example. The explicit definition of gates may be omitted for messages with unambiguous names in an MSC. The environmental merge operator then identifies the equally named messages to, respective from, the environment in both MSCs.

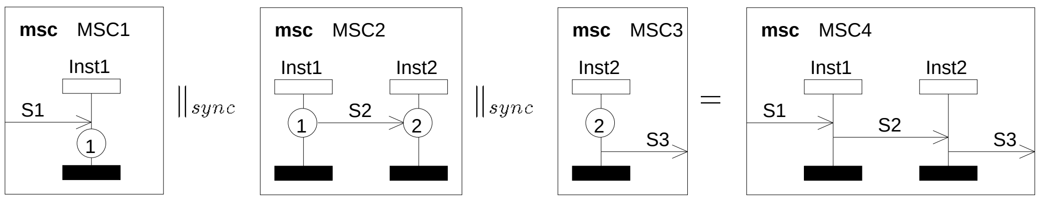
Similarly to interworking merge, the synchronisation merge ($\|_{syn}$) of two MSCs is their interleaved composition with the restriction that the MSCs are forced to synchronise on a



(a) Example for an environmental merge



(b) Example for a synchronisation merge



(c) Example for a synchronisation condition merge

Figure 1. Composition operators for MSCs

set of communication actions. The set consists of the communication actions concerning every pair of entities which the MSCs have in common. Figure 1 (b) presents an example. The synchronisation merge is an adaptation of interworking merge [7] to asynchronous communication.

Synchronisation condition merge ($\|_{sync}$) means that the synchronisation is performed with respect to connectors, i.e., distinguished synchronisation points. Connectors are represented graphically by annotated circles. Figure 1 (c) presents an example. This solution is closest to the original example presented in October 94 at the ITU-T SG 10 meeting [8] and also closest to conditions. It also has the greatest flexibility with respect to composition but probably is less transparent than, e.g., $\|_{syn}$.

2.2. Object orientation in MSC

OO-techniques in MSC may help to emphasize similarities in different MSCs, facilitate the reuse of complete MSC diagrams or parts of them, and support the structuring of complete MSC specifications or individual diagrams. The discussions on object orientation in MSC are related to the introduction of the concepts *type*, *inheritance*, and *virtuality* in the MSC language. We mainly follow the ideas and notations in [4–6].

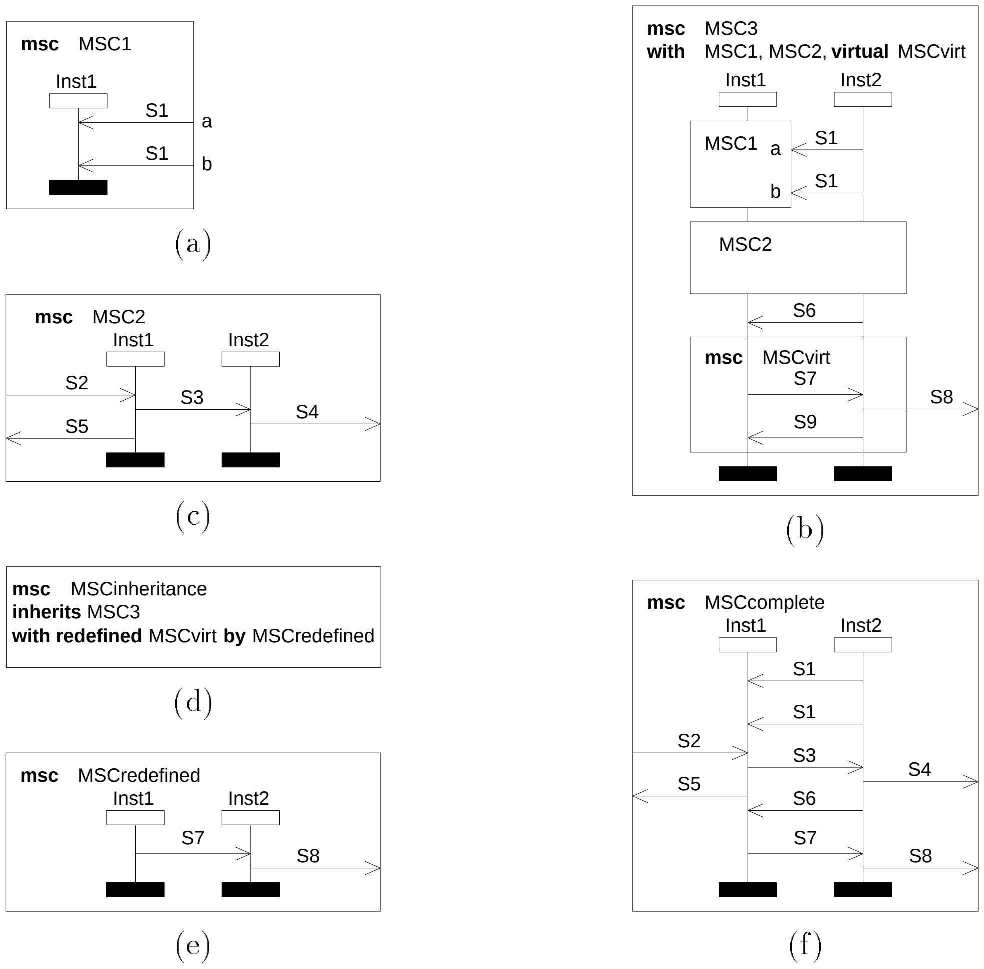


Figure 2. Object-oriented concepts and MSC

Each MSC can be seen as a definition of an MSC type. MSC types can be used in other MSC types. Figure 2 (a) shows the MSC type definition *MSC1*. The identifiers *a* and *b* at the frame of *MSC1* are *gates*. Gates are used to define the connection points when an MSC type is used in another type. For example, the MSC type *MSC3* in Figure 2 (b) uses the MSC type *MSC1* and *MSC2*. The identifiers within the reference symbol are connected to the gate definitions in (a) by name identification.

MSC3 also refers to *MSC2* in (c). *MSC2* exchanges messages with its environment which are not described in *MSC3*. By convention the environment of the used type is connected with the environment of the using type, i.e., in our case the messages *S2*, *S4*, and *S5* are also part of message exchange of *MSC3* with its environment.

For practical reasons sometimes it may be useful to define a new MSC type directly inside another one. For this purpose [6] introduces *inline definitions*. Figure 2 (b) provides an example. *MSC3* includes the *inline definition* of the MSC type *MSCvirt*.

An MSC may use behaviour which is specified in another MSC. This can be done by *inheritance*. For example, the MSC *MSCinheritance* in Figure 2 (d) inherits the signal exchange specified in (b).

An MSC type can refer to *virtual types*. Virtual means that it is possible to adapt an MSC to special configurations or situations by redefining virtual types. For example, the MSC type *MSC3* in Figure 2 (b) includes the inline definition of the virtual type *MSCvirt*. *MSCinheritance* in Figure 2 (d) inherits *MSC3* and refers to the redefinition *MSCredefined* of *MSCvirt* which is shown in (e). This redefinition means that within *MSCinheritance* the behaviour of *MSCvirt* is replaced by *MSCredefined*.

The main advantage of the MSC language, compared to other trace languages, is the ability to describe system behaviour in a clear, graphical and intuitive way. The introduction of object oriented concepts in MSC may lead to situations where the understanding of a simple MSC description becomes rather complicated. Figure 2 provides an example. The behaviour described by the MSC *MSCcomplete* in (f) is not very complicated. It is identical to the behaviour defined by the object-oriented MSC *MSCinheritance* in (d). But *MSCinheritance* refers directly and indirectly to four other diagrams shown in (a), (b), (c) and (e). However, additional design rules might help to keep the MSCs understandable.

3. Application to a telecom example

In this section we introduce the proposed example and apply both composition operators and OO-techniques in order to describe the example in an adequate manner.

3.1. The EWSD example

The example was provided by the public switching systems division of the Siemens AG. It concerns a small extract of the highly distributed EWSD switching system. Therefore we refer to the example as *EWSD example*. An EWSD system consists of one central processor and many group processors. We distinguish between peripheral message flow, i.e., message exchange between group processors and system environment, and internal message flow, i.e., message exchange among group processors and central processor.

For a given internal message flow the peripheral message flow shows several variations depending on the supported signalling system. Each group processor may support ten signalling systems. Among them are Signalling System No. 5 (CCITT Rec. Q.120-180 Blue Book), Signalling System No. 7 (CCITT Rec. Q.721-725 Blue Book), and ISUP (CCITT Rec. Q.730, Q.741, Q.761-766 Blue Book).

In our EWSD example the internal message flow is specified by an initial MSC which is an abstract model of the real message flow. In particular, the peripheral message flow does not correspond to any concrete signalling system and can be considered as an abstract representation showing typical features of the various real signalling systems. The real signalling systems are introduced afterwards in form of peripheral message flow variants. These are provided in form of separate MSCs and can be combined independently with the internal message flow.

Our example refers to a behaviour of an EWSD system configuration with one central processor *CP* and two group processors *GP/A* and *GP/B*. The initial MSC of the EWSD example is shown in Figure 3. We selected the peripheral message flow variants of the signalling systems ISDN PA, ISDN BA and ISUP for describing the independent combination of peripheral message flow variants.

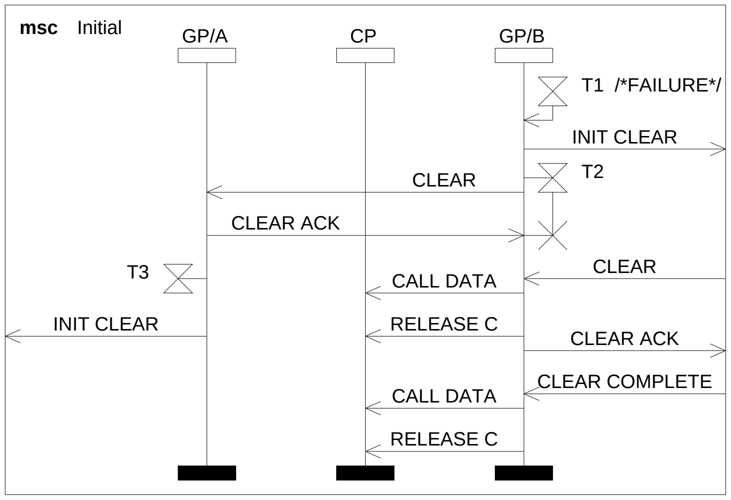


Figure 3. The initial MSC

3.2. Applying composition operators

Within the EWSD development process special message flow diagrams are employed. For this paper, only the message flow of these special diagrams is taken into account without simplifications whereas we abstract from EWSD specific constructs.

This means for applying composition operators the original EWSD specific diagrams are transformed into the Z.120 form and, where necessary, enhanced with gates (cf. Figure 4) and connectors (cf. Figure 6). The MSCs in Figure 6 are closest to the original EWSD diagrams. The column instances include connections denoting the causal dependencies between the different events [9,10].

The environmental merge of peripheral and central parts is described by using the MSCs in the Figure 4. Since the central communication part described in the initial MSC remains the same for all peripheral variants it is extracted from the initial MSC in separated form (cf. Figure 4 (a)). Each variation of the peripheral part of group processor A - corresponding to the signalling systems ISDN PA and ISUP (cf. Figure 4 (b) and (c)) - can be combined with each peripheral part of group processor B - corresponding to the signalling systems ISDN PA and ISDN BA (cf. Figure 4 (d) and (e)) - whereby the central part always remains the same. We obtain the following four descriptions:

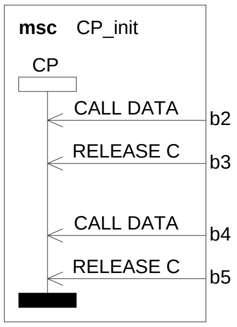
$$(1) \quad A\_PA \parallel_{env} B\_PA \parallel_{env} CP\_init$$
$$(2) \quad A\_PA \parallel_{env} B\_BA \parallel_{env} CP\_init$$
$$(3) \quad A\_ISUP \parallel_{env} B\_PA \parallel_{env} CP\_init$$
$$(4) \quad A\_ISUP \parallel_{env} B\_BA \parallel_{env} CP\_init$$

By using an additional **or** operator the four expressions can be combined to the single expression:

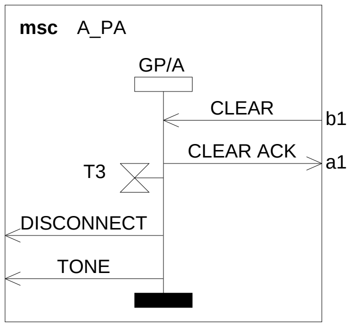
$$(A\_PA \textbf{ or } A\_ISUP) \parallel_{env} (B\_PA \textbf{ or } B\_BA) \parallel_{env} CP\_init$$

The main idea for synchronisation merge is the same as for environmental merge. Therefore, it is sufficient to demonstrate the merging operation for one signalling system only, namely for ISDN PA. By using the MSCs in Figure 5 we obtain the following expression:

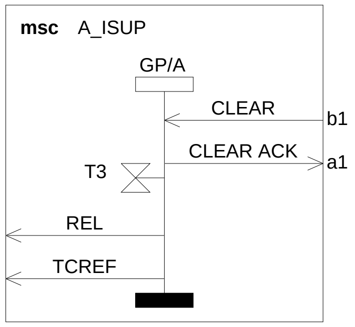
$$A\_PA\_syn \parallel_{syn} B\_PA\_syn$$



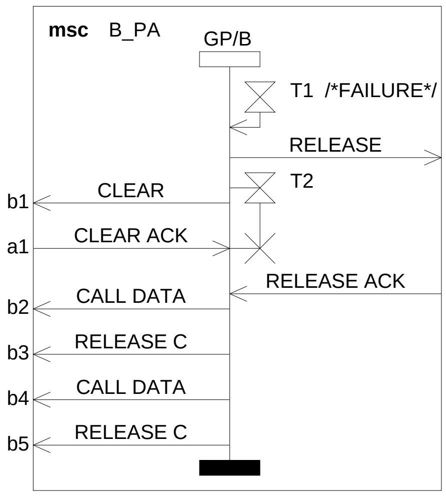
(a) Central processor



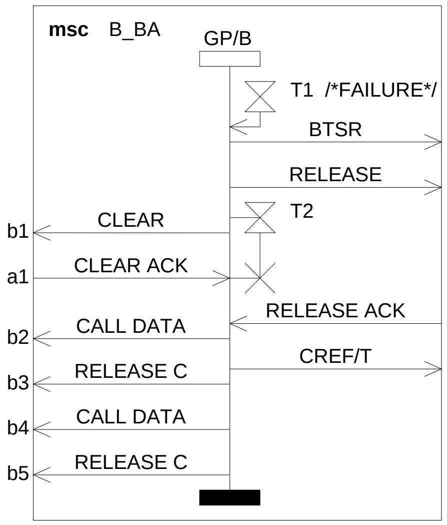
(b) GP/A: ISDN PA



(c) GP/A: ISUP

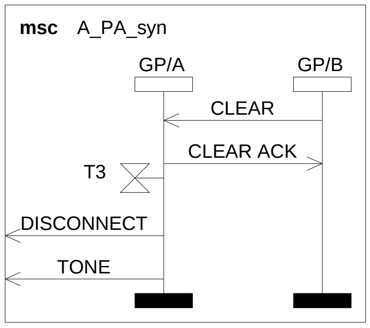


(d) GP/B: ISDN PA

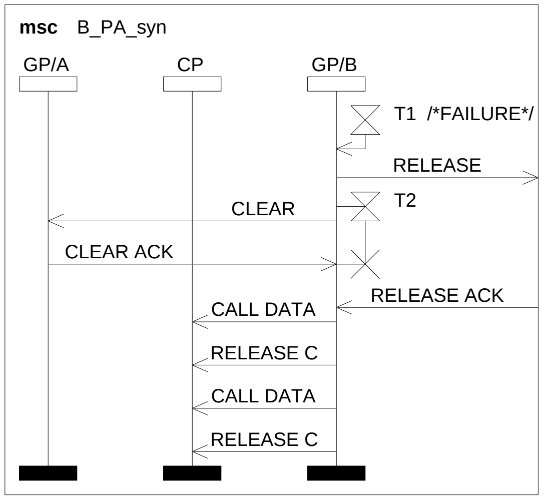


(e) GP/B: ISDN BA

Figure 4. Behaviour of central processor CP and group processors GP/A and GP/B

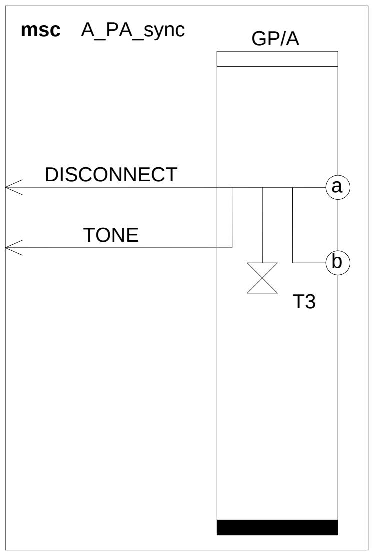


(a) ISDN PA behaviour of GP/A

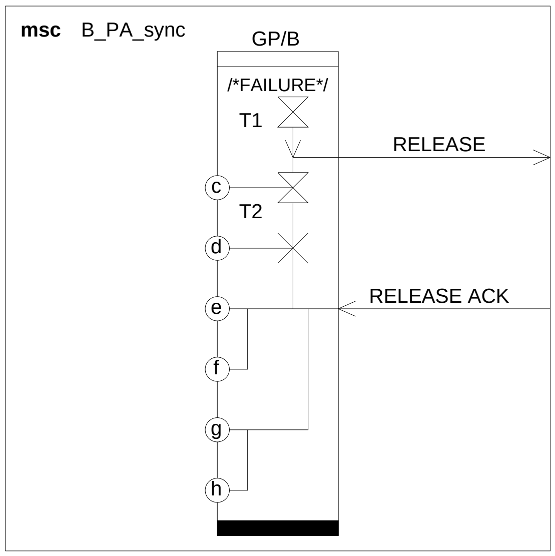


(b) ISDN PA behaviour of GP/B

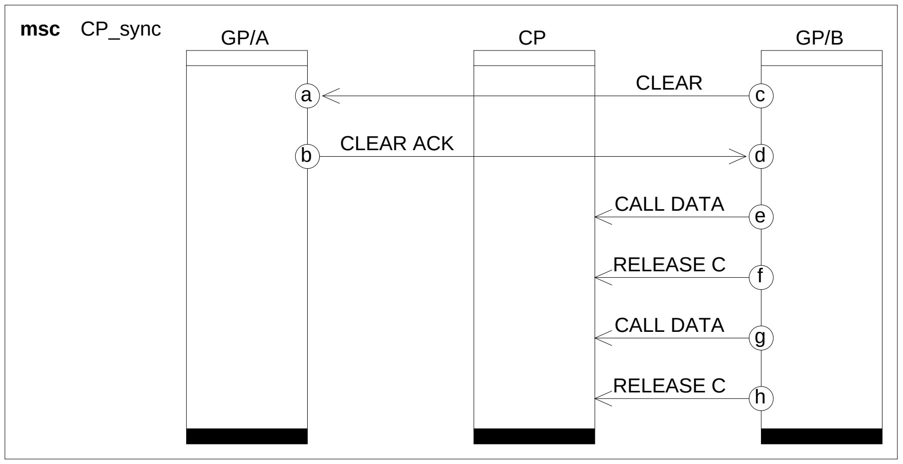
Figure 5. MSC diagrams for modelling synchronisation merge



(a) ISDN PA behaviour of GP/A



(b) ISDN PA behaviour of GP/B



(c) System internal behaviour

Figure 6. MSCs for modelling synchronisation condition merge

In comparison with the application of the environmental merge the communication with the central processor is integrated in the MSC describing the behaviour of GP/B (cf. Figure 5).

The synchronisation condition merge operation also is demonstrated for ISDN PA only. By using the MSCs in Figure 6 we obtain the following expression:

$$A\_PA\_sync \parallel_{sync} B\_PA\_sync \parallel_{sync} CP\_sync$$

3.3. Applying object-oriented techniques

There are two possibilities to apply object-oriented techniques, i.e., types, inheritance, and virtuality (cf. Section 2.2), to the EWSD example. One possibility is to combine MSC type definitions to new MSC types, each describing one trace of the EWSD system. A second possibility is to use inheritance and virtuality. Virtual parts of an abstract MSC

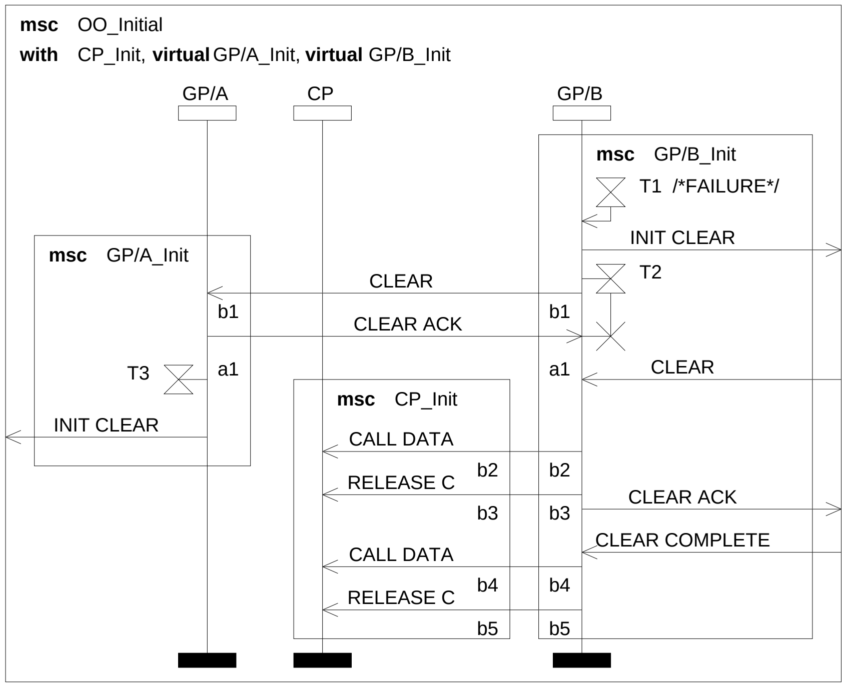


Figure 7. An object-oriented modification of the initial MSC

type are redefined when a concrete system run of the EWSD system is specified. The building blocks for an object-oriented modelling of the EWSD example are a modification of the initial MSC and the alternative behaviours of the group processors.

A modification of the initial MSC (cf. Figure 3) is shown in Figure 7. Compared with the original initial MSC the object oriented modification *OO_Initial* includes three inline MSC type definitions. The MSC types *GP/A_Init* and *GP/B_Init* are declared as virtual types, i.e., they can be redefined when the initial MSC is adapted to different system configurations.

The parts of the initial MSC *OO_Initial* which are adapted to the different system configurations concern the behaviour of the group processors GP/A and GP/B. The possible alternatives are defined in Figure 4 (b) – (e). Compared with the MSC standard Z.120 these MSCs only include additional gate definitions, but no special object-oriented concepts. The MSC descriptions in Figure 8 refer to these MSCs.

The object-oriented MSC descriptions of the EWSD example are based on the MSCs in the Figures 4 (b) – (e) and 7. Their combination leads to four different behaviours. Applying OO techniques may lead to the four MSCs shown in Figure 8 (a) – (d).

The MSCs in Figure 8 (a) and (b) are assembled from MSC types only. From a more abstract point of view this modelling can be seen as a graphical description of the environmental merge (cf. Sections 2.1 and 3.2); except that the gate connections between the three MSC types are established explicitly within the diagram and not implicitly by name identification. Furthermore, the use of an additional **or** operator provides the possibility to describe all possible combinations in a graphical, simple and intuitive manner. This is shown in Figure 8 (e).

The MSCs in Figure 8 (c) and (d) inherit the initial MSC. By reference the two virtual

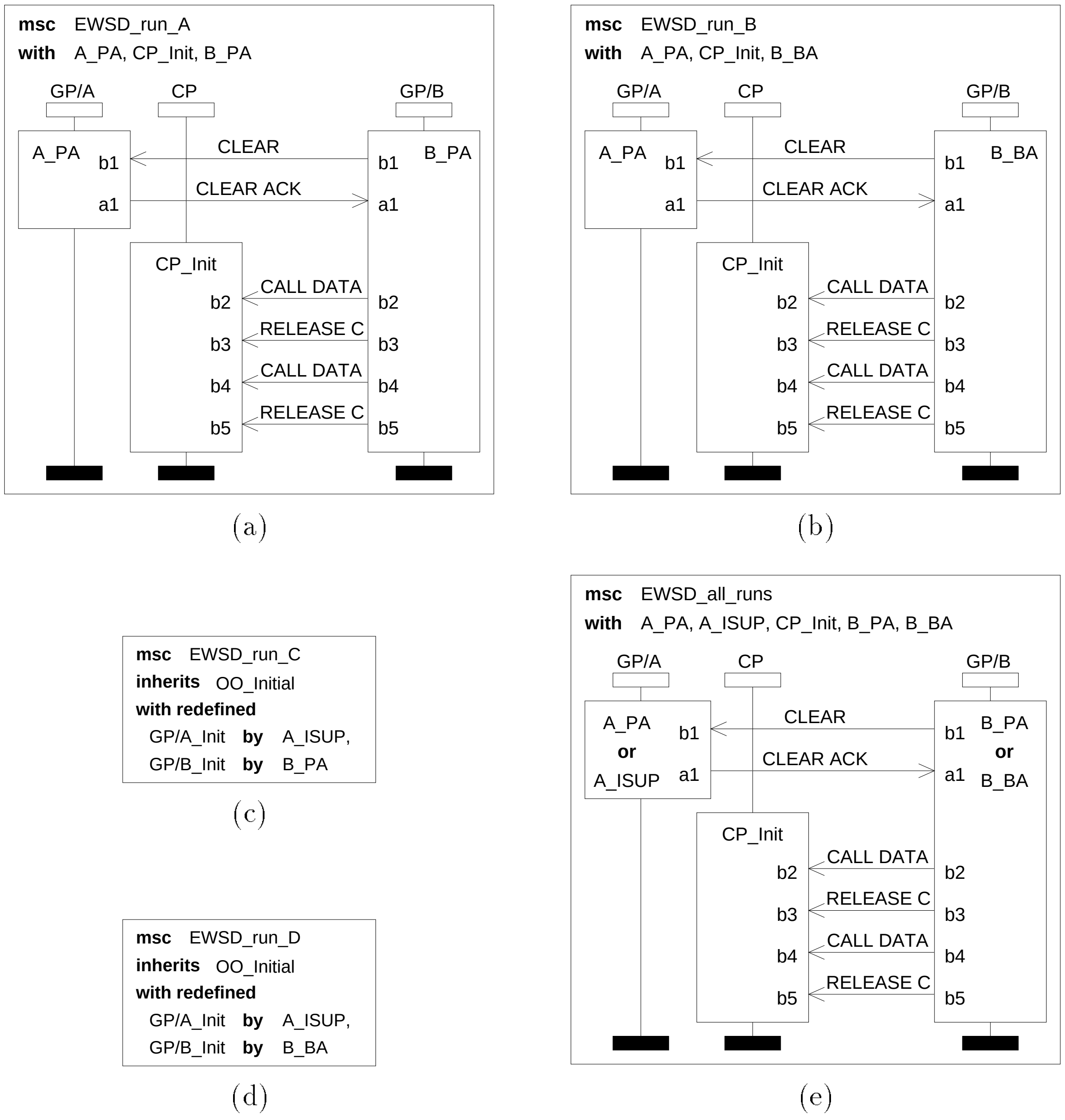


Figure 8. Object oriented MSC descriptions of the EWSD example

MSC type definitions *Init_GP/A* and *Init_GP/B* of the initial MSC (Figure 7) are redefined: *Init_GP/A* by *A_ISUP* of Figure 4 (c), *Init_GP/B* either by *B_PA* of Figure 4 (d) or *B_BA* of Figure 4 (e).

4. Discussion

In the previous section two alternative but nevertheless related modelling techniques have been confronted: Composition techniques and OO-techniques. In this final discussion, advantages and disadvantages of both techniques collected as experienced during the analysis of the EWSD example shall be summarized.

4.1. Advantages and disadvantages of composition techniques

The EWSD example shows that the proposed MSC composition techniques provide a very powerful means for combining MSCs in a compact and elegant manner. By using additional constructors, like the '**or**' in Section 3.2, complete MSC specifications can be described by means of only one expression.

Looking at the EWSD development process, environmental merge and synchronisation condition merge seem to be closest to the current use of the EWSD specific diagrams, i.e., peripheral variations are described in a similar manner as shown in the Figures 4 and 6.

Main problem of such composition techniques is that the overview about the complete system may easily get lost. A good compromise seems to be the synchronisation merge (cf. Sections 2.1 and 3.2): Within synchronisation merge, all instances are included in one MSC which are necessary to describe the message flow without splitting messages into send- and reception events. Thus, the context remains constantly visible. Synchronisation merge has been used within Phillips/PKI to solve the horizontal paging problem [7].

One step towards a more user friendly representation would be the development of a graphical representation of composition operations instead of abstract textual formulas. Several proposals have been made into this direction. Generalization of MSC overview diagrams [11,3], the tree notation in GEODE [2,1]; and, as described in Section 3.3, the combination of MSC types are all very promising.

4.2. Advantages and disadvantages of OO-techniques

The OO-techniques, as they are used here, have the advantage that MSCs which belong together are integrated within a larger frame from the very beginning. Whereas pure composition techniques may easily end up in something like a puzzle, OO-techniques could be compared with a large painting where all parts remain integrated within the context.

The MSC type concept, although types are not characteristic to object-orientation, turned out to be very promising.[1] The use of types within other types can be viewed as a special graphical representation of the environmental merge. In this context additional constructors, like the **or** in Figure 8 (e), may allow a compact and intuitive description of complete MSC specifications.

Characteristic OO-techniques like inheritance and virtuality seemed to be less fruitful for the EWSD example. Using the proposed concepts for inheritance, virtuality, and redefinition [5,6] one would get a rather large number of MSCs with redefined parts. For each variant one needs a different description. Furthermore, MSCs as shown in Figure 8 (c) and (d) are neither graphical nor intuitive.

5. Outlook

The EWSD example clearly demonstrates that advanced structural concepts for MSCs are necessary for a specification of modern telecom systems. The development and investigation of composition techniques and OO-techniques is a central goal of the present ITU study period 1993 - 1996. In particular, parallel merging operations have to be developed

[1]We would like to mention that there is some ongoing discussion in ITU-T SG 10 Q.9 on the replacement of the *submsc* construct by a more general MSC type concept.

based on solutions proposed so far. Abstract composition formulas should be replaced by graphical notations. Therefore we started to experiment with generalized MSC overview diagrams. The MSC type concept seems to be very useful and should be elaborated for integration into the revised Z.120. According to this case study, the use of MSC types can be viewed as a graphical representation of the environmental merge. Typical OO-techniques, i.e., inheritance and virtuality, need further studies before conclusions can be drawn. For individual MSCs at least, these concepts seem to be less appropriate. It should be discussed whether it is more fruitful to declare complete MSCs as virtual (instead of sub-diagrams) and to apply inheritance to whole MSC documents or sub-documents.

Acknowledgements

The elaboration of this paper was funded partially by the KWF-Project No. 2555.1 *'Graphical Methods in the Test Process'*. We would like to thank Jürgen Hennicke and Hanspeter Ruckstuhl from the public switching systems division of the Siemens AG for providing the EWSD example and for valuable discussions.

REFERENCES

1. V. Encontre. MSC Extensions/Restrictions. Reference Manual – GEODE Simulator, Chapter 6, Verilog SA, Toulouse (France), 1993.
2. V. Encontre, E. Delboulbe, P. Gavaud, and A. Boussalem. Combining Services, Message Sequence Charts and SDL: Formalism, Method and Tools. In O. Faergemand and R. Reed, editors, *SDL'91 Evolving Methods*. North-Holland, 1991.
3. J. Grabowski, P. Graubmann, and E. Rudolph. Towards an SDL-Design-Methodology Using Sequence Chart Segments. In O. Faergemand and R. Reed, editors, *SDL'91 Evolving Methods*. North-Holland, 1991.
4. O Haugen. MSC - Type Concept. ITU-T SG 10 Experts Meeting in Darmstadt (Germany), March 1993.
5. O Haugen. MSC - Type Concept. ITU-T SG 10 Meeting in Geneva (Switzerland), October 1993.
6. O Haugen. MSC - Structural Concepts (& Discussion). ITU-T SG 10 Rapporteurs Meeting in Turin (Italy), April 1994.
7. S. Mauw, M. van Wijk, and T. Winter. Syntax and Semantics of Synchronous Interworkings. In O. Faegemand and A. Sarma, editors, *SDL'93 - Using Objects*, October 1993.
8. E. Rudolph. Case Study on MSC Structural Concepts. ITU-T SG 10 Meeting in Geneva (Switzerland), October 1994.
9. E. Rudolph. Generalized Causal Ordering. ITU-T SG 10 Rapporteurs Meeting in Turin (Italy), April 1994.
10. Swiss PTT. Weakening the time ordering along MSC instances (Proposal for discussion). Delayed Contribution D.94-X/3, CCITT SG X (WP X/3) Interims Meeting in Geneva, November 1992.
11. Z.100 I (1993). SDL Methodology Guidelines. Appendix I to Z.100. ITU-T, July 1993.
12. Z.120 (1993). Message Sequence Chart (MSC). ITU-T, September 1994.