Towards an SDL-Design-Methodology Using Sequence Chart Segments

Ekkart Rudolph, Peter Graubmann
Siemens AG München, ZFE IS SOF
Otto-Hahn-Ring 6
D-W8000 München 83
rudolph@ztivax.uucp / gr@ztivax.uucp

Jens Grabowski
Universität Bern
Länggassstrasse 51
CH-3012 Bern
grabowsk@iam.unibe.ch

Abstract: Composition mechanisms for Sequence Charts are investigated aiming at a comprehensive system description. Within a systematic composition method standard basic building blocks for Sequence Charts, called Sequence Chart segments, are defined analogously to (Petri net) process segments. The usefulness of SC-segments for SDL-requirement engineering, system construction, analysis, simulation, and test case generation is pointed out.

1 Introduction

At the last SDL-Forum Lisbon 1989 signal flow diagrams with SDL-specific extensions have been introduced called Extended Sequence Charts (ESCs) [GraRu 89]. Within this contribution in particular the role of ESCs in the software development process has been exhibited. In this paper now an idea is worked out how to put SDL-requirement engineering using Sequence Charts (SCs) into a more formal framework. Some advantages of such a methodology are described in chapter 3.

The main importance of Sequence Charts lies in the stage of requirement definition. However in practice only relevant, representative, exemplary traces can be specified by means of SCs without composition mechanisms. For a complete behaviour description an enormous amount

This work was partially performed within the ESPRIT projects ATMOSPHERE (No. 2565) and REX (No. 2080).

of SCs would be needed. Within the SDL Methodology Guidelines [Bel 91] therefore LOTOS was suggested as an alternative for the specification of interfaces between blocks. Consequently, composition operators analogous to LOTOS have been suggested for the extension of SCs [ESC 90].

By means of global conditions (initial, intermediate, or final) another possibility for the composition of SCs has been suggested within the Message Sequence Chart standardization. A final or intermediate condition determines the possible continuations by other SCs. Each continuation is determined by identification of final and initial conditions. The composition by global conditions still is rather restrictive. Therefore, it has been suggested to introduce local conditions [MSC 91].

Several investigations concerning the relationship between SDL and Petri nets already exist [Grab 90]. Within these investigations the analogy between SDL-systems and Petri net systems on the one hand, and the analogy beween Sequence Charts and Petri net processes on the other hand has been pointed out. A composition method using local conditions already has been suggested for Petri net processes by means of *process segments* [Chong 85][1]. Process segments are basic behaviour structures of Petri net processes from which all possible Petri net processes can be composed. It suggests itself to carry over this idea to Sequence Charts. We call the basic behaviour structures of SCs *Sequence Chart segments* (SC-segments).

In chapter 2 an informal motivation for SC-segments is presented. In chapter 3 the role of SCs and SC-segments within the software development process is discussed. A formal definition of process segments within Petri net theory is presented in chapter 4. In chapter 5 an analogous definition for SC-segments is provided. Finally in chapter 6 an example is worked out.

2 Informal description of Sequence Chart segments

2.1 The idea of Sequence Chart segments

All possible behaviours of an SDL-system can be described by traces. Regarding the set of all possible traces of a finite system one observes that in general it turns out to be infinite and that a single trace may be arbitrarily long. This fact is mainly due to cyclic system behaviour.

It is characteristic for systems containing cycles that the complete behaviour may be composed from partial behaviour descriptions. A certain behaviour pattern may occur in different traces or several times within one trace. Such partial behaviour descriptions, which are in fact segments of traces, shall be used as building blocks from which the entire set of traces may be constructed.

Of course all traces (presented in form of Sequence Charts) may be composed from the state transitions of activated system processes. However, such a composition procedure is not reasonable since coherence of behaviour patterns is lost and the system structure is no longer visible for the designer.

[1]Originally in [Chong 85] *process period* is used instead of *process segment*. We choose *segment* to avoid misunderstandings we observed occasionally.

Larger building blocks are demanded. We aim at a definition of *inseparable* trace segments which are "as long as possible" (the meaning of which will be defined below) and from which the system behaviour can be composed. *Inseparable* in this context means that these trace segments are never split by others. The set of these building blocks has to be maximal in the sense that all possible traces can be constructed out of its elements and minimal in the sense that no element of the set can be composed from others in this set.

The building blocks we aim at can be characterised in the following way: they are (a) (inseparable) sequential trace segments, (b) (inseparable) cyclic trace segments and (c) sections of sequential and (d) cyclic trace segments in which other trace segments are embedded. This is illustrated in figure 2.1 where the reachability graph of a system is indicated schematically (the numbered circles refer to global system states). In this example, parallelism is hidden in global state transitions.

A simple comparison of the trace segments in figure 2.1 with the reachability graph shows that all possible traces can be composed from the presented segments. This construction procedure can be represented in form of a graph (figure 2.2).

We would like to point out that the figures 2.1 and 2.2 merely serve as an illustration of our ideas. For nontrivial systems the situation is not that simple. Furthermore, the starting point of our considerations is not a case where the reachability graph for the generation of characteristic trace segments is given. On the contrary, the system requirements are defined by means of the specified characteristic trace segments.

(Extended) Sequence Charts are particularly suitable for the description of traces in SDL-systems. They offer an appropriate graphical representation of traces which also presents the time (causal) ordering of events in an intuitive manner. SCs also suit for the description of the characteristic trace segments of an SDL-system. Such an SC represents a *period of time* within the possible traces of an SDL-system. In order to exhibit this particularity, we use the notation *Sequence Chart segments* for those SCs.

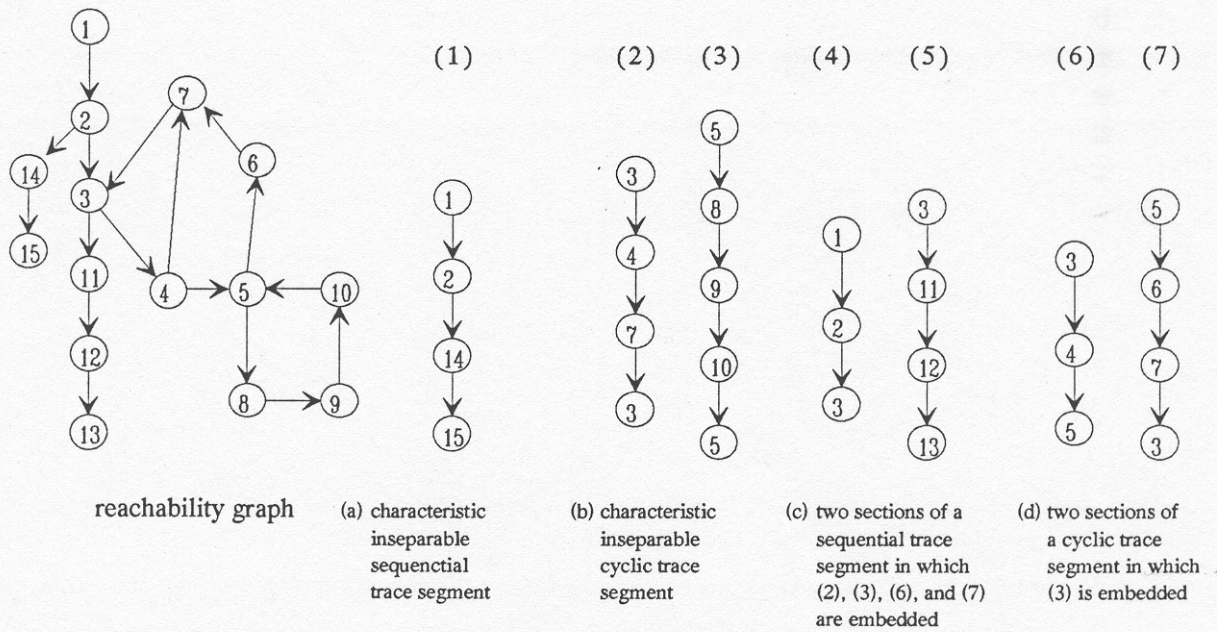

(1)
(2)
(3)
(4)
(5)
(6)
(7)
reachability graph
(a) characteristic inseparable sequenctial trace segment
(b) characteristic inseparable cyclic trace segment
(c) two sections of a sequential trace segment in which (2), (3), (6), and (7) are embedded
(d) two sections of a cyclic trace segment in which (3) is embedded

Figure 2.1: Reachability graph with 7 characteristic trace segments

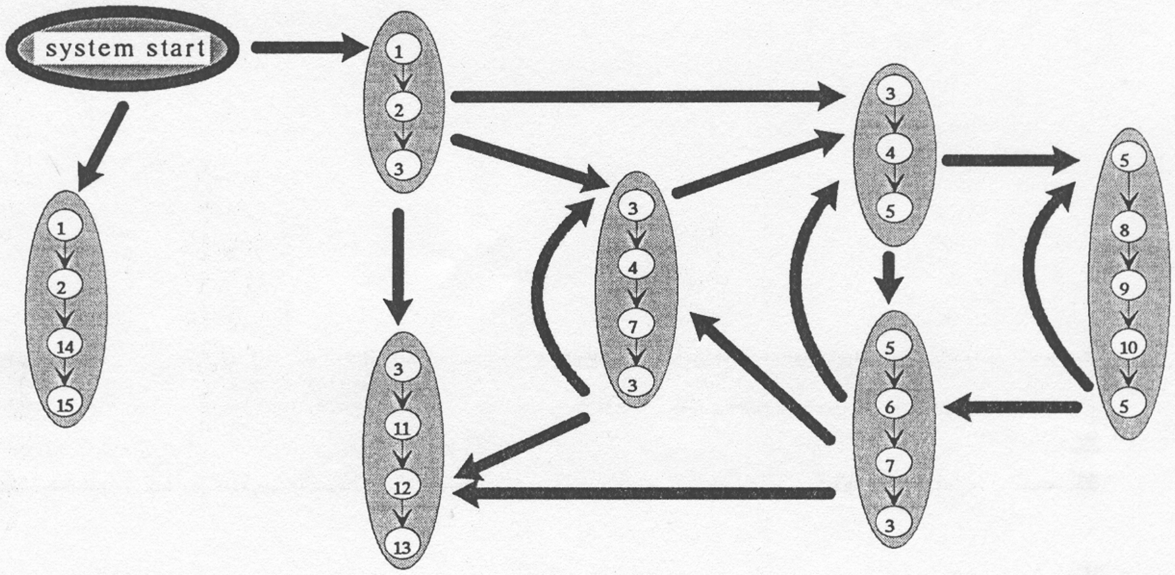


Figure 2.2: A graph describing the composition of all possible traces from the trace segments of figure 2.1

Before we introduce a formal definition of process segments and Sequence Chart segments (chapter 4 and 5), we want to point out the difference between Petri net processes and Sequence Charts on the one hand and between process segments and Sequence Chart segments on the other hand.

2.2 Differences between Sequence Charts and Petri net processes

A Petri net process (PN-process) describes the behaviour of a Petri net system (PN-system) as a trace in form of a so-called occurrence net which is itself a specialized Petri net. The relation between both is set up by a mapping from the PN-process to the underlying PN-system. This mapping is defined in such a manner that the structure of PN-process and underlying PN-system are consistent. Furthermore, it is assumed that the initial state and all subsequent states of the process are mapped onto reachable states of the underlying PN-system (cf. chapter 4).

Sequence Charts and PN-processes are conceptually analogous but differ by the fact that the mapping between a SC and the underlying SDL-system is set up only partially. This is due to the fact that in early stages of design the knowledge about the system is not yet complete, since the SDL-system has first to be constructed from the given set of SCs.

From this point of view the development of an SDL-specification from a set of SCs consists in the construction of a suitable mapping from the given SCs to the SDL-system. One possible approach towards the definition of such a mapping has been described in [GraRu 89] by means of a stepwise enhancement of SCs by SDL-symbols.

Additional information, however, is needed about the set of traces from which the complete system behaviour has to be (re)constructed. On the Petri net side this is provided in form of a set of process segments. In order to handle this problem also for SDL-systems and SCs we define *Sequence Chart segments*. The automatic generation of SC-segments from incomplete sets of SCs in addition allows to analyse the behaviour of the system specified up to this stage.

2.3 Differences between Sequence Chart segments and process segments

Process segments are introduced within Petri net theory exclusively employing local conditions. By that, they describe *global* and *local* (partial) system traces. Contrary to that, SC-segments use both local and global conditions. All initiated (SDL-) processes are involved in a global system trace. Correspondingly, within a local system trace only a partial set of all initiated (SDL-) processes contributes.

The global conditions in Sequence Charts correspond to special sets of places in occurrence nets, so-called *B-cuts*. The initial states for (local and global) process segments are such B-cuts (or rather certain sub-sets of them).

The strategy pursued in our approach is to describe global system runs by means of global conditions and to use local conditions for a later insertion of local traces. The exclusive use of local conditions would force system designers to work in early stages of design already on a low level of abstraction, i.e. with fine-grained refinements. On later stages, however, it may be necessary to identify local conditions within global conditions or to split global conditions into local conditions. We cannot go further into these problems within this paper.

In figure 2.3 (a) a (global) trace of the Inres service [Hog 91] is described. This trace contains local and global conditions. Figure 2.3 (c) shows how the (local) cycle described in 2.3 (b) can be inserted within this global trace. Finally, it should be noted that the formal description of the construction procedure for a graph as shown in figure 2.2 from local traces may become very complex. Therefore we restrict ourselves to a precise definition of SC-segments.

2.4 Related descriptions and analysis methods

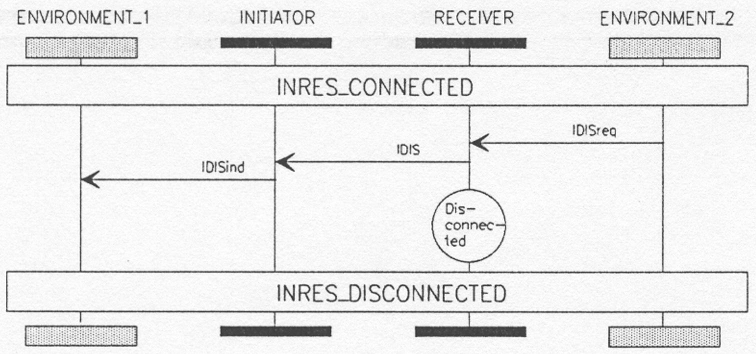
The ideas pursued in this paper originate from Petri net theory in which the system (re)construction from a given behaviour description has been investigated already in some detail. Because of the well established analogies between Petri nets and SDL-specifications on the one hand, and between Petri net processes and Sequence Charts on the other hand the concept of (Petri net) process segments is carried over to the SDL-world by introducing SC-segments.

The relationship between SDL and Petri nets has been investigated mainly because analysis methods from Petri net theory can be utilized for SDL-specifications. In particular, reachability analysis and the calculus of invariants has been carried over to the SDL-world. The reachability graph in principle describes the complete system behaviour in form of system states and state transitions. In practice, its exploitation often fails due to state explosion. The same holds for the Asynchronous Communication Tree (ACT) [Hog 88].

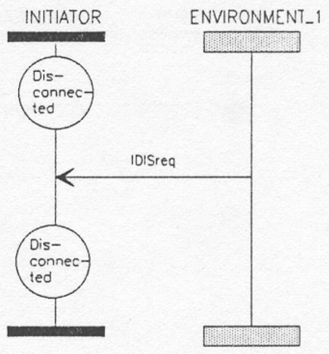
Petri net invariants, particularly transition invariants, have been employed successfully for the analysis of SDL-specifications [GrauRu 85]. Transition invariants prove to be useful for the analysis of cyclic system behaviour. By means of realization of minimal support invariants the elementary (nonreducible) cyclic system traces may be constructed.

Petri net processes describe the causal structure of nets as Sequence Charts do for SDL-systems. Thus, we envisage an improvement of analytical power by combining reachability analysis, transition invariants, and analysis techniques used for process segments.

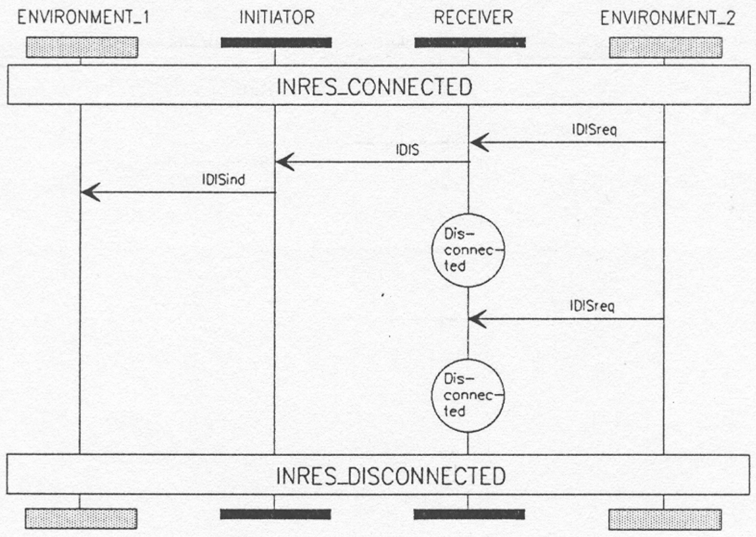
As indicated in figure 2.2, the system behaviour can be described by means of the set of segments analogously to the reachability graph. Trace cycles obtained from minimal support invariants are closely related to process segments. The clear representation of causality within Petri net processes and Sequence Charts is carried over to segments since they describe fragments of them. The relations pointed out in this chapter are illustrated in figure 2.4 .



(a) Global trace of the Inres example with *global* and *local* conditions



(b) Local cycle of the Inres example with *local* conditions



(c) Possible combination of (a) and (b)

Figure 2.3: The combination of local and global trace segments of the Inres example

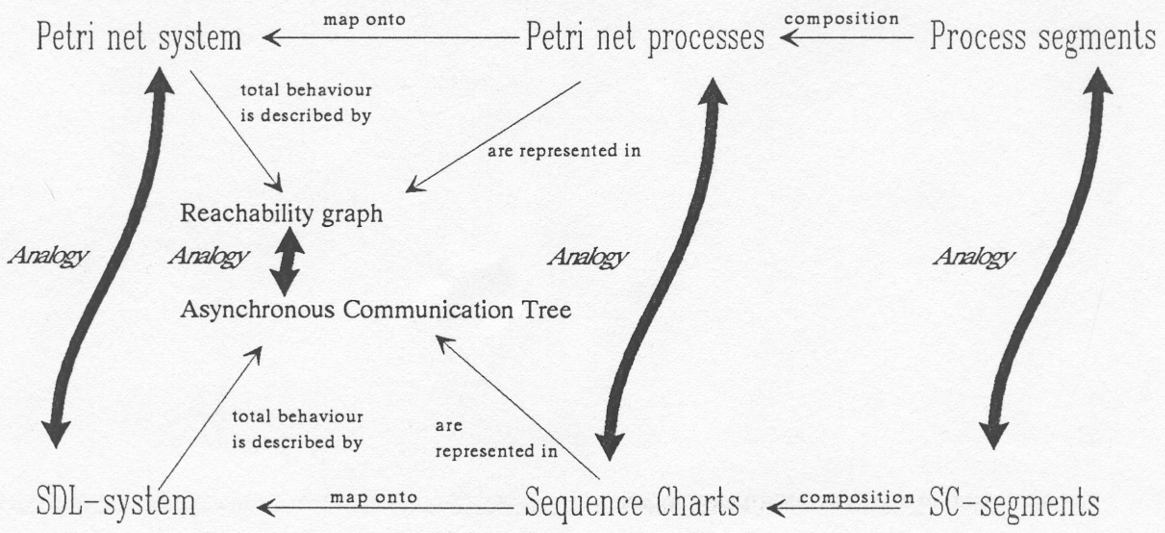


Figure 2.4: Relations between SDL and Petri nets

3 Sequence Chart segments and their role in system engineering

3.1 The role of SCs and SC-segments in system specification

Sequence Charts are widely used in practice for the specification of the system behaviour within requirement engineering. Compared with other trace languages SCs have the advantage of a clear graphical layout and structuring. The shortcomings, namely the lack of composition operations and the consequent limitations, have been discussed several times in literature. In particular, LOTOS has been proposed as an alternative for the specification of block-interfaces [Bel 91].

In order to overcome these limitations the proposal for the Message Sequence Chart standardization has been enhanced by the introduction of global and local conditions [MSC 91]. Global and local conditions admit the composition of comprehensive SCs from SC-building blocks. The choice of such SC building blocks is left to the system designer. No unnecessary restrictions shall be prescribed. The only restriction consists in the mandatory specification of local or global conditions defining the composition rules. However, special building blocks, namely SC-segments, may be singled out.

The idea is to generate the SC-segments automatically which should be possible at each stage of system specification. Thus, SC-segments represent standard building blocks which may be carried separately and in parallel with the specified SC fragments.

The set of generated SC-segments represent the desired system behaviour. Beyond that, the computer aided composition of SC-segments offers the possibility of an immediate simulation of the specified behaviour. Viewing segments as the smallest units admits the abstraction from details of communication and the analysis of the *essential* behaviour structures of the system.

3.2 Sequence Charts, Sequence Chart segments and SDL

The introduction of local conditions within the standardization of Message Sequence Charts (MSCs) was demanded also in order to make a systematic construction of SDL-specifications from MSCs feasible [KDD 90]. As it was pointed out in this contribution, local connectors define the relationship between MSCs and SDL process diagrams.

The definition of SC-segments is not limited to MSCs. The same composition and decomposition method by means of SC-segments as basic units may be carried over to Extended Sequence Charts (ESCs). In particular SC-segments may be defined for the special variants of ESCs, the *State-form* and the *State Input-form*, as defined in [GraRu 89]. Thus, the stepwise refinement of SC-fragments, especially of SC-segments, by means of SDL-Symbols can be used to generate (more or less) complete SDL process diagrams as described in [GraRu 89].

3.3 The use of Sequence Chart segments in test case generation

Recently Sequence Charts have been proposed for a support of test case generation [Hog 90]. In this approach, SCs have been used for a selection of test cases. Automatic test case generation from state machines suffers from the enormous amount of resulting test cases. SCs are suggested as a means for reduction of generated test cases to the *interesting* and *critical* ones. Because of the complexity of the Asynchronous Communication Tree (ACT) which is used for the generation of test cases - the complexity is known for the reachability graph of Petri nets - SCs can provide an interesting alternative representation in trace form. By means of SCs the time (causal) ordering of events is presented in an intuitive compact graphical manner which has obvious advantages compared with the interleaving representation within the ACT. At least SCs may be used as a supplement. In any case, the mentioned shortcoming of SCs should be overcome by SC-segments in the direction of a comprehensive description.

4 Definition of process segments based on Petri nets

4.1 Petri net processes and the properties of their building blocks

The idea of Sequence Chart segments (SC-segments) as a tool for compact behaviour definition can be traced back to the notion of process segments of Petri nets which first were introduced operationally in [Chong 85] in which they are defined as the results of a certain (unfortunately slightly incorrect) algorithm. We provide here a formal definition of process segments based on their desired properties and the behaviour semantics of Petri nets. Thus, we gain insight into the structure of process segments. This information about their nature is exploited for the formal definition of SC-segments in the next chapter.

The behaviour of a Petri net system is given as a set of Petri net processes (PN-processes) each of which can be interpreted as the history of an actual *run* of the system. A PN-process thus reflects one of the possible system behaviours and, consequently, conflicts are already resolved. An orthogonal cut through a PN-process resembles a snapshot of the PN-system's behaviour: it shows which system transitions (system actions) and which system states co-exist or happen concurrently. A longitudinal section through a PN-process informs about what actions and system states are performed respectively produced in sequence. Altogether, a PN-process describes the partial ordering of the various occurrences of PN-system actions and PN-system states. They are denoted by so called occurrence nets with an additional labelling function into places and transitions of the underlying Petri net system, thus establishing the behaviour semantics of the net. Thereby, occurrence nets are defined to be cycle-free Petri nets that never branch at places.

PN-processes can be of arbitrary length due to a cyclic system behaviour. Furthermore, concurrency of Petri net systems forces the description of mutually independent parts of its behaviour into one process, blurring thus a natural distinctive feature of system analysis. Process segments now shall serve as a means to clarify the system structure in terms of behaviour (or PN-process) building blocks.

PN-processes can be constructed systematically out of Petri net systems, but process segments are gained from the set of PN-processes. In the sequel we give a definition of process segments based on PN-processes; an algorithm for the determination of process segments however computes them out of the underlying Petri net system.

(1) Process segments are fragments of PN-processes, i.e. each process segment can be identified in at least one PN-process.

(2) Process segments are connected, i.e. they do not split into two (or more) parts without arcs crossing from one part to the other.

(3) No prefix of a process segment is a process segment itself. (A prefix denotes - in terms of behaviour history - the same past though the observation was stopped earlier.)

(4) Process segments may be cyclic, but they do not contain proper sub-cycles.

(5) Process segments are *linked structures*. This property is the crucial one in our context and will be discussed in the next paragraphs.

Requiring process segments to be linked structures shall provide - intuitively spoken - that there is no system behaviour (i.e. no PN-process) in which one can identify a process segment broken into two parts which occur separated by a cyclic behaviour sequence. This gives the gist of the definition, however, it is but a first intuitive approximation which excludes too many cases. Hence in greater detail now:

> A PN-process fragment is called *linked*, iff no matter how it is cut into two pieces along concurrent system states, these two pieces are inseparable in all fragments of PN-processes.
>
> Two such pieces q_1 and q_2 are called *inseparable* in a PN-process fragment *p*, iff the following holds:
>
> if there is a possibility to cut the PN-process fragment *p* into three pieces along concurrent system states such that
>
> - q_1 occurs at the end of the first piece,
>
> - q_2 occurs as a prefix of the last, and

- both q_1 and q_2 do not occur in the cyclic middle part r,

then this middle part r does not really separate q_1 and q_2, i.e. q_1 concatenated with q_2 (in sign: $q_1 \oplus q_2$) is either prefix or end piece of the PN-process fragment $q_1 \oplus r \oplus q_2$.

The above list of five properties provides the basic selection criteria for process segments. A twofold selection process will eventually produce them. The first step is to determine the so-called *pre-segments*.

4.2 Selection of pre-segments

A closer investigation into linked structures shows that prefixes of linked structures are also linked. Thus the criterion (3) helps to establish the desired maximality of process segments. Yet under certain circumstances, it may happen that arbitrary iterations of cyclic linked structures are still linked. Hence, criterion (4) helps to restrict process segments to finite fragments of PN-processes. Both examples show that we have to establish a sophisticated selection process which distinguishes between too long and too short PN-process fragments.

We therefore define a set of PN-process fragments which shall be called *pre-segments*, because we will later choose the process segments among them. To determine the pre-segments is in itself again a three-step process:

(a) Let us at first collect all PN-process fragments which are connected as well as linked structures and for which additionally holds that they contain at most one sub-cycle which in any case must be placed at the end of the fragment.

This first selection step looks rather involved but mirrors exactly a simple construction procedure to gain those PN-process fragments. The request for the final sub-cycle translates for this algorithm into a mere termination condition. One hardly can do better: During a local construction process a cycle can be detected not earlier until one realizes that the cycle is closed. In a next step,

(b) let us discard all PN-process fragments which are prefixes of other PN-process fragments in this set.

By this step, we particularly get rid of PN-process fragments with uncompleted final sub-cycles. But by now, we have eliminated certain PN-process fragments we still need. Yet, during the following step we get them back again:

(c) Let us prune off all final sub-cycles in the remaining PN-process fragments.

With these three steps we get the set of pre-segments as a basis for the selection of process segments. An algorithm for the calculation of pre-segments clearly would perform these three steps intertwined. It also would perform the selection of process segments out of pre-segments interleaved with the above steps. But here are now the criteria for this selection process:

4.3 The determination of process segments

The set of pre-segments is in accordance with the above formulated list of five properties for process segments, but it is still too large. It would allow to construct not only the set of PN-processes for the given Petri net system but rather the set of PN-processes starting with an arbitrary marking provided that this marking is reachable from the given initial marking for which the pre-segments have been calculated. This is an advantage for system analysis, however, it makes it still necessary to select the appropriate minimal number of process segments which are sufficient to construct all possible PN-processes. This selection process is driven only by comparing and matching the pre-segments' initial and final places (i.e. places without incoming respectively outgoing arcs) and it comprises two rules:

(*) A pre-segment with its initial places mapped wholly into the initial marking of the respective Petri net system is a process segment.

(**) A pre-segment with its initial places mapped wholly into a union of the images of final places of some process segments is a process segment itself.

This selection process preserves and produces respectively the required properties of process segments: The set of pre-segments, and hence, the set of process segments are finite. Each process segment itself is of finite length (because process segments do not contain sub-cycles). Each process of the Petri net in question can be considered a composition of process segments, where only at the end of the process a prefix of a process segment may be used.

5 Definition of Sequence Chart segments

The analogy between Sequence Charts (SCs) and PN-processes [Gra 90] admits translation of the defining properties for (Petri net) process segments to Sequence Chart segments (SC-segments). However, there are some subtle differences which have to be taken into account as was pointed out already in chapter 2.3.

The definition of PN-processes assumes the existence of a mapping of occurrence nets onto an underlying Petri net system. The starting point for the definition of SC-segments is somewhat different. We assume a given set of SCs containing global and local conditions which admit SC-composition and decomposition according to the MSC-composition-rules [MSC 91]. Since the definition of SC-segments should be possible at each stage of the system development we do not assume an underlying SDL-system to which the SCs refer.

In fact, the given set of SCs from which the SC-segments shall be constructed may be rather rudimentary. Thus, SCs may be interpreted immediately as occurrence nets, the labelling of its elements (reflecting the mapping), however, is less straightforward. In order to employ the methods from PN-processes nevertheless, we abstract from a mapping onto an underlying system and interprete the labelling as an identification of certain elements.

It turns out that only global or local conditions in SCs admit such an identification according to their semantics definition [MSC 91]. Message names cannot be related precisely to the labelling of occurrence nets since they do not refer to events in the sense of Petri nets (which have to be named differently in different states) but to a higher level of abstraction (actions).

As a consequence the following definition of SC-segments is based exclusively on global and local conditions within the prescribed set S of SCs and the corresponding (de)compositions.

We define a set S' of Sequence Chart fragments which corresponds to the set of process fragments of chapter 4 and from which SC-segments can be derived analogously to process segments from process fragments:

S' contains all possible SC-sections which can be obtained from S by composition and decomposition of SCs. According to [MSC 91] initial, intermediate, and final conditions may be attached to SCs. Sequence Charts may be decomposed at intermediate conditions. SCs without intermediate conditions cannot be decomposed further. In this sense they provide the *atoms* of S'. Contrary to process fragments (cf. chapter 4) these SC-atoms are no more split. SC-segments are built out of these SC-atoms and form a distinguished kind of SC-molecules.

Keeping in mind that composition and decomposition of SC's refers to these *atoms* the definitions from chapter 4 can be adapted. SC-composition is denoted by $\oplus$.

(1) SC-segments are elements of S'.

(2) SC-segments are connected.

(3) No prefix of a SC-segment is a SC-segment.

(4) SC-segments may be cyclic but they do not contain proper subcycles.

(5) SC-segments are linked.

A Sequence Chart SC of S' is linked iff for all possible decompositions of SC in SC_1 and SC_2 both parts appear to be inseparable:

SC_1 and SC_2 are *inseparable* in SC, iff the following holds:
If SC can be decomposed into three pieces SC_i, SC_m, and SC_2 such that

- $SC = SC_i \oplus SC_m \oplus SC_2$,
- SC_m neither contains SC_1 nor SC_2, whereby
- $SC_i = SC_{i'} \oplus SC_1$ ($SC_{i'}$ may be empty)

then SC_m does not really separate SC_1 and SC_2
but is either concurrent to SC_1 or SC_2, i.e.:

$$SC_{i'} \oplus SC_1 \oplus SC_m \oplus SC_2 = SC_{i'} \oplus SC_1 \oplus SC_2 \oplus SC_m$$

or

$$SC_{i'} \oplus SC_1 \oplus SC_m \oplus SC_2 = SC_{i'} \oplus SC_m \oplus SC_1 \oplus SC_2$$

By means of these five properties SC-segments can be selected analogously to chapter 4.2. The explicit definition of an initial state corresponding to an initial marking is not necessary. There is also no further selection corresponding to PN-segments from pre-segments.

6. An example (Inres service)

As has been pointed out in chapter 3.1 the specification process shall not be restricted by the concept of SC-segments. At each stage of design the SC-segments may be generated automatically from the specified SC-traces. The idea is that the system designer initially specifies the essential traces, i.e. the traces which define the main purpose of the system. In figure 6.1 this is described for the Inres service [Hog 91]. The possible connection points between SC-segments are represented by global and local conditions. Exceptional or error cases are added afterwards. Cf. figure 6.2 and 6.3 which show the possible disconnection cases in different global states and figure 6.4 where the rejection of a connection request is displayed.

At each stage of design the already defined SC-segments may be composed to obtain the current total system behaviour. This is demonstrated in figure 6.5 with respect to the SC-segments defined in figure 6.1 - 6.4. Note, that the global state INRES_WAIT_FOR_resp indicates neither the beginning nor the end of an SC-segment. The enhancement of segments by means of SDL-symbols (figure 6.6) admits the generation of SDL-process diagrams showing the behaviour prescribed by these segments.

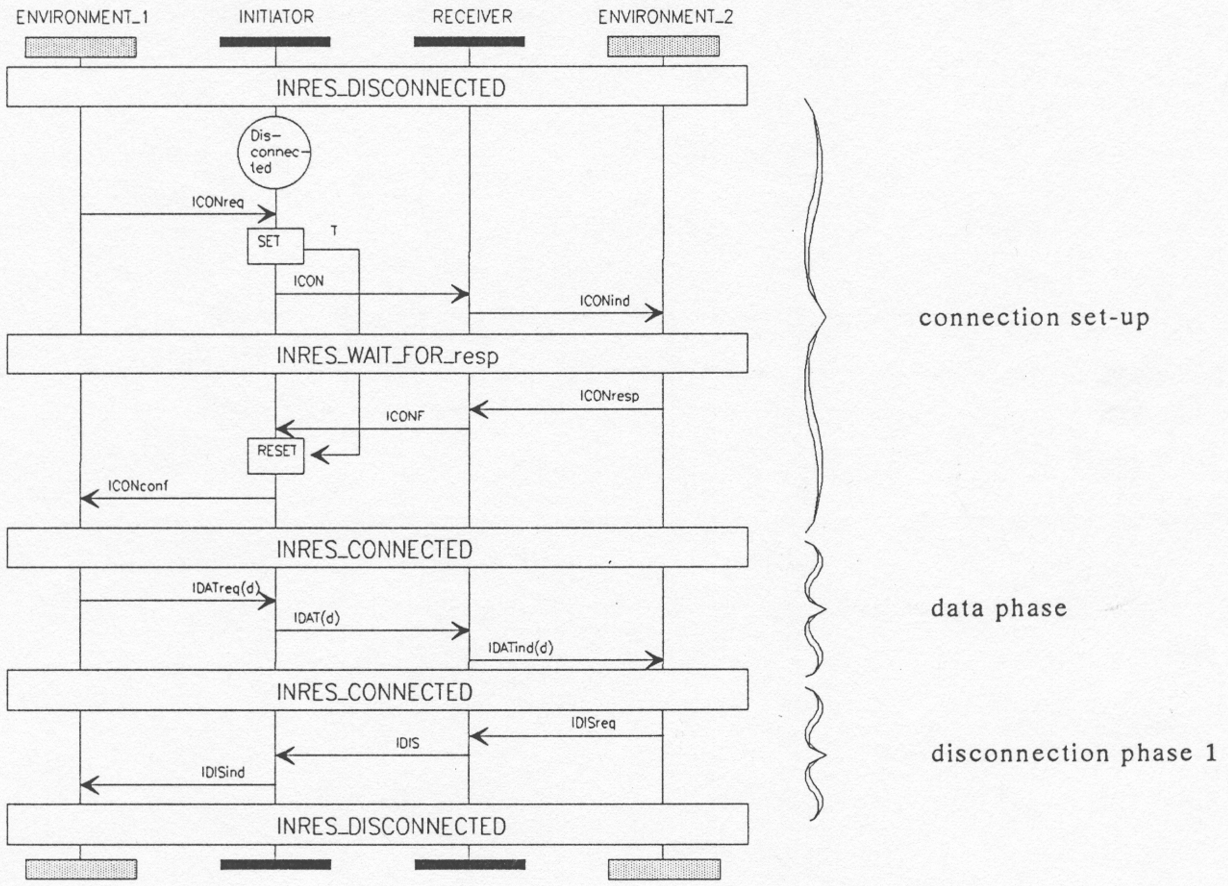


Figure 6.1: Global Inres trace with 3 indicated Sequence Chart segments (named: *connection set-up*, *data phase* and *disconnection phase 1*)

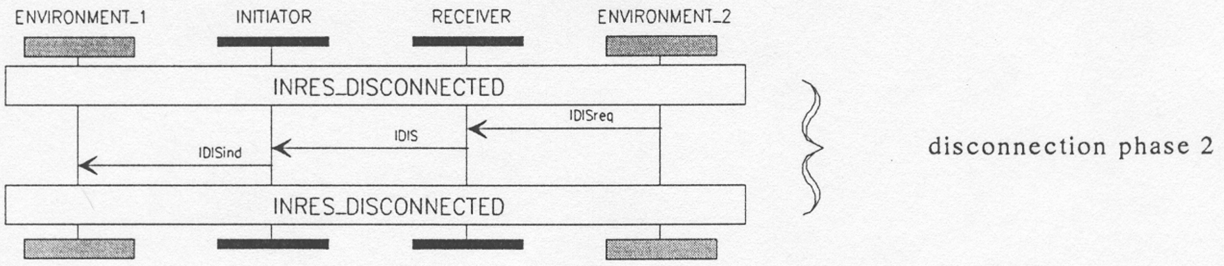


Figure 6.2: Global Inres trace with disconnection in an INRES_DISCONNECTED-state, one SC-segment is indicated (named: *disconnection phase 2*)

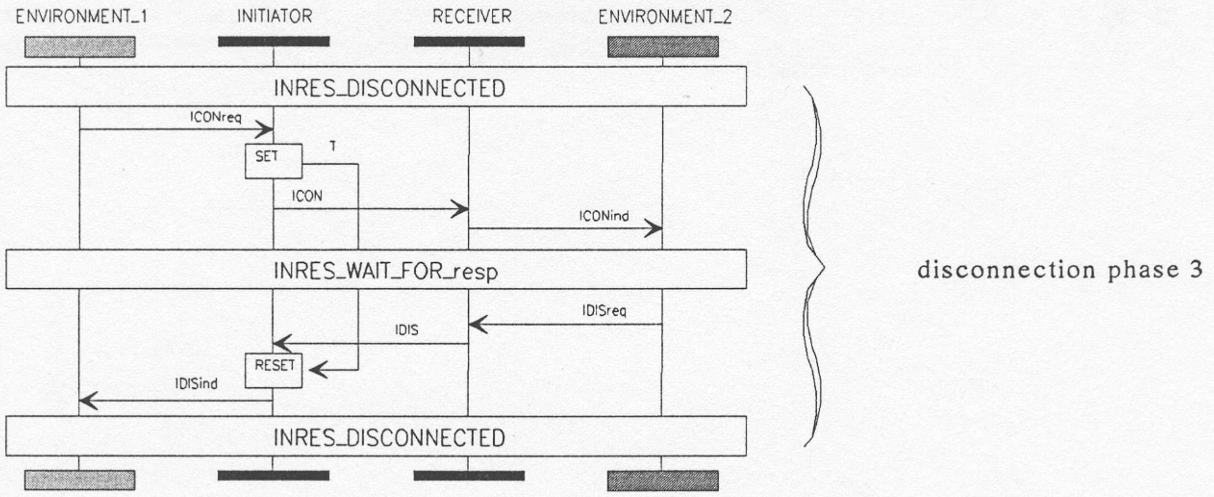


Figure 6.3: Global Inres trace with disconnection in an INRES_WAIT_FOR_resp-state, one new SC-segment is indicated (named: *disconnection phase 3*)

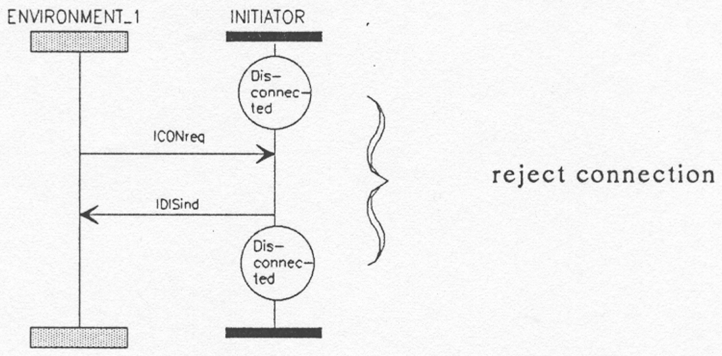


Figure 6.4: Local Inres trace with rejection of a connection request, one SC-segment with local conditions is indicated (named: *reject connection*)

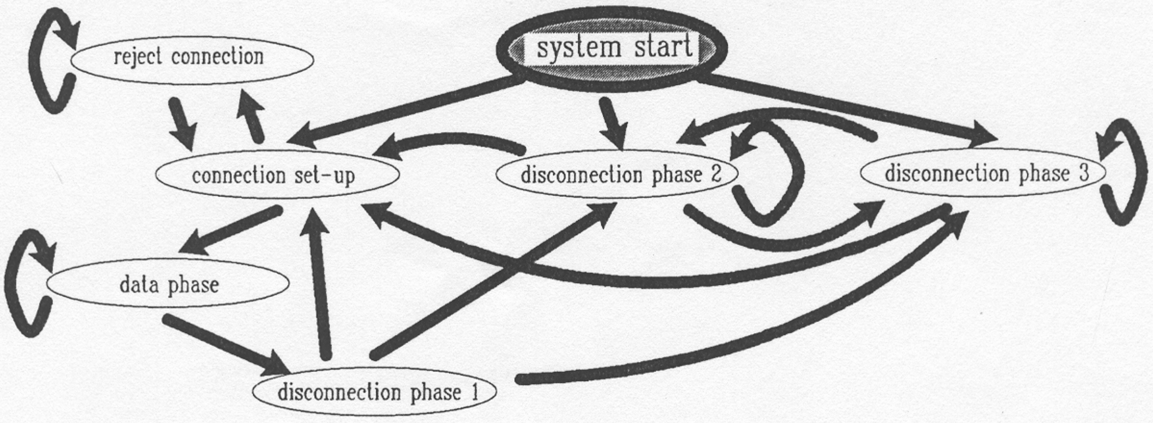
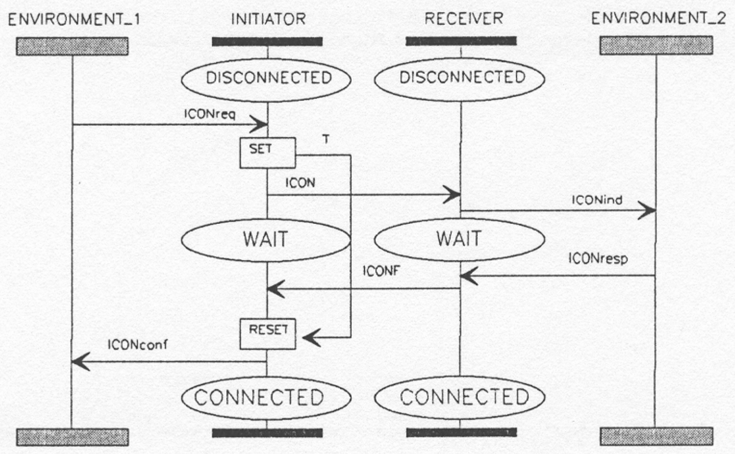
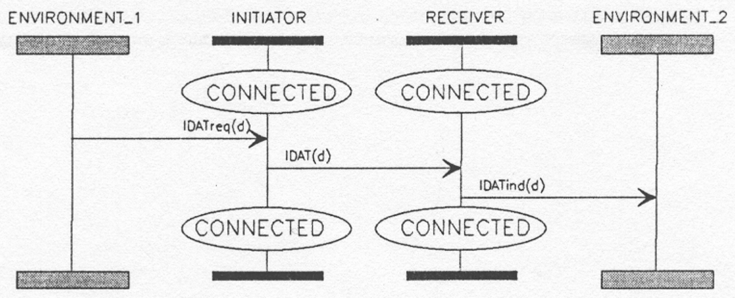


Figure 6.5: Composition rules for the SC-segments indicated in figure 6.1 - 6.4 (inscribed cycles describe SC-segments)



(a) Enhancement of SC-segment *connection set-up* with SDL state symbols (global system states are left out)



(b) Enhancement of SC-segment *data phase* with SDL state symbols (global system states are left out)

Figure 6.6: Enhancement of Sequence Chart segments with SDL state symbols

Literature

[Bel 91]	Belina, F.: SDL Methodology Guidelines CCITT- meeting Geneva 1991
[Chong 85]	Chong-Yi, Y.: Process Periods and System Reconstruction Lecture Notes in Computer Science 222, Advances in Petri nets 1985, Edited by G. Rozenberg, Springer Verlag 1986
[ESC 90]	Rudolph, E.: Proposal for the Standardization of Extended Sequence Charts CCITT experts meeting Copenhagen 1990
[Gra 90]	Grabowski, J.: Statische und dynamische Analysen für SDL-Prozess- diagramme auf der Basis von Petri-Netzen und Sequence Charts University of Hamburg, Diploma thesis, April 1990
[GraRu 89]	Grabowski, J.; Rudolph, E.: Putting Extended Sequence Charts to Practice SDL '89 The Language at Work - O. Faergemand and M. M. Marques (Editors), North-Holland 1989
[GrauRu 85]	Graubmann, P.; Rudolph, E.: A Method and a Tool for the Validation of SDL-Diagrams Second SDL Users and Implementers Forum, Helsinki 1985
[Hog 88]	Hogrefe, D.: Automatic Generation of Test Cases from SDL Specifications CCITT SDL Newsletter 12, 1988
[Hog 90]	Hogrefe, D.: Conformance Testing Based on Formal Methods Invited presentation, Forte'90, Madrid, 1990
[Hog 91]	Hogrefe, D.: OSI Formal Specification Case Study: the Inres Protocol and Service Technical report, University of Berne, April 1991
[KDD 90]	KDD contribution to WP X/3: The Importance of State Descriptions in Sequence Charts CCITT experts meeting, Turin 1990
[MSC 91]	Rudolph, E. : Message Sequence Chart CCITT-meeting Geneva 1991
[Rei 85]	Reisig W.: Petri Nets EATC Monographs on Theoretical Computer Sciences, Vol.4, Springer 1985