The ETSI SDL model for the Intelligent Network Application Protocol

Dieter Hogrefe
ETSI TC MTS chairman
Institut fuer Telematik, Germany
hogrefe@itm.mu-luebeck.de

Jan Ellsberger
ETSI PEX competence center, France
jan.ellsberger@etsi.fr

Abstract

This paper describes the development of the ETSI INAP SDL model, one of the first complete and strictly formal SDL specifications making use of full object orientation. The model is used as an integral part of the ETSI and ITU-T standards for intelligent network and is used within ETSI to produce the corresponding test suite in a semi-automatic way.

Keywords

Intelligent network, INAP, SDL, MSC, TTCN, test case generation, object orientation

1 INTRODUCTION

With a complete SDL model for the Intelligent Network Application Protocol ETSI is exploring new grounds. Traditionally the specifications published by ETSI make use of SDL [1] only in an informal and illustrative way. This has advantages, e.g. understandability and development time, but also disadvantages, e.g. the specifications are not machine processable.

The SDL work was done in the ESTI Sub-Technical Committee SPS3 on a voluntary basis with support of the Permanent Expert Group (PEX) at ETSI and the Technical Committee Methods for Testing and Specification (TC MTS). The SDL was developed in parallel with the INAP protocol standardization with only a little delay. The work started in the middle of 1995 and the SDL model was finished in the middle of 1997 at the same time with the publication of the INAP protocol CS2. See [3] for CS1 and [4] for CS2.

Unlike other SDL models for INAP, the ETSI model has been done in very close cooperation with the standardization process and is now published together with the standard as a normative annex A of [4].

The purpose of the complete SDL model for INAP is the facilitation of

- service development
- feature interaction studies
- switch design
- test case generation

In the past, in particular the development of test suites based on a protocol or service specification proved to be difficult. The reasons are diverse. One reason is that the test suite in many cases comes very late, a too long time after the respective protocol or service specification. Therefore it is only of limited value for the industry which makes products based on a particular specification. The products are already on the market before the test suite is readily published. The other reason is cost. Because of the risc that the value is limited in many cases, the motivation of the companies to participate in the development of a test suite voluntarily is sometimes low. This means that ETSI has to set up project teams in the costed work program in order to develop the test suite.

The development of a complete SDL model as the normative part of the protocol or service specification is an attempt to tackle these problems.

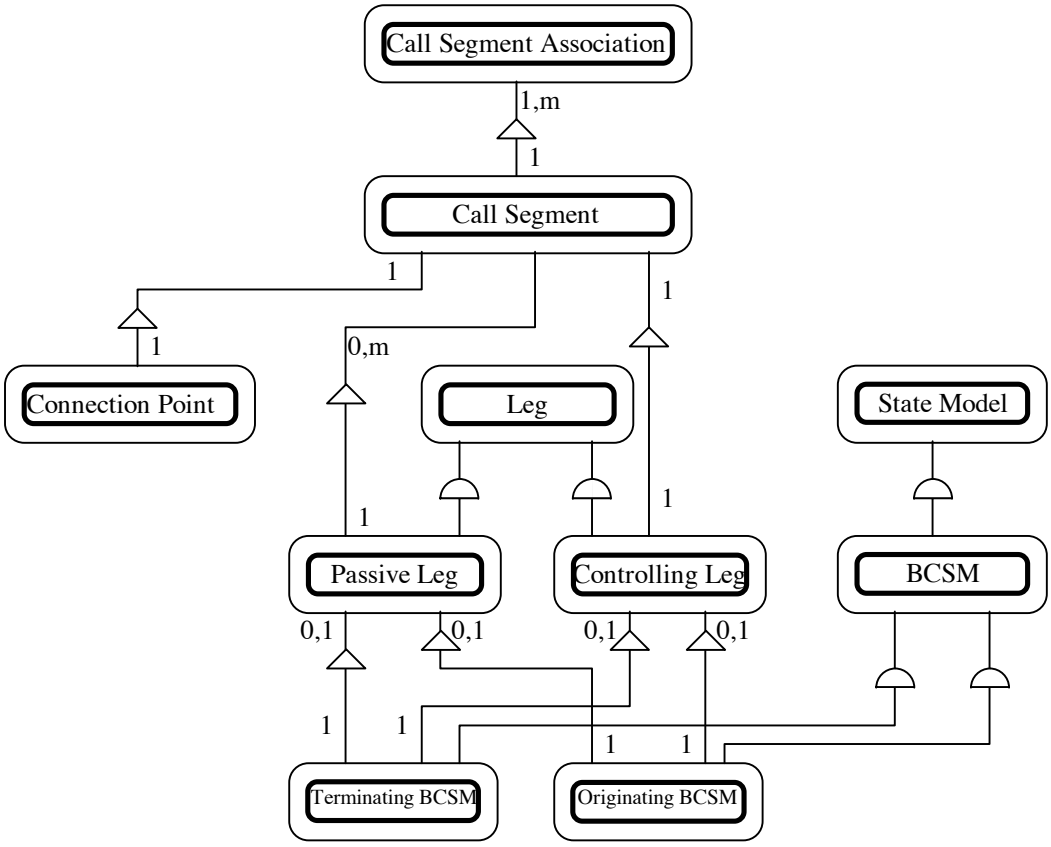
2 ARCHITECTURE OF THE MODEL

The INAP CS1 and CS2 models are specified with SDL 96 [5] in an object oriented way. The CS2 specification inherits the CS1 and only specifies the difference between CS1 and CS2. Consequently the architectures of CS1 and CS2 are more or less the same.

The SDL model specifies precisely and unambiguously the behaviour and the interworking between the different functional entities: CS, CSA, SSF-FSM, BCSM. The data structures are not completely specified. They are included in [4] and appropriate references are made in the SDL model.

The model issues a plattform for service emulation and the development of test cases based on IN services.

Fig. 1 shows the INAP CS1/CS2 information model. There are one or many Call Segments in one Call Segment Association. There is a one to one correspondence between Call Segment and Connection Point, and so on. There are two objects of type BCSM, the OriginatingBCSM and the TerminatingBCSM.



F

ig.1 IN CS1/CS2 information model

Fig. 2 shows the SDL model of CS1 SSF/CCF. The CS2 equivalent is shown in Fig. 5 from where it can be seen that the overall structure is the same for both capability sets. CS2 inherits most of the items of the CS1 model and refines them. The objects in the information model of Fig. 1 map to the SDL model of Fig. 2 in the following way.

The Call Segment Association object is a process type in the SDL model. The CallSegmentAssociation manages the
- the creation of CallSegments and
- the dialogue with the SCF.

The Call Segment object is divided into two process types in the SDL model: CallSegment and SSF-FSM. The process type CallSegment manages the
- identifiers of the legs (connection view). This data structure models the Connection Point object.
- creation of the O-BCSMs and T-BCSMs.
- creation of the SSF-FSM.

- filtering of detection points
- processing of connection view oriented IN operations

The process type SSF-FSM manages the
- processing of IN operations
- handling of detection points (EDPs and TDPs).

The Connection Point object is modeled by a data structure as described above.

The Leg object is also a data structure within the process type CallSegment. It is identical to the data structures of Passive Leg and Controlling Leg. The data structure contains the status of the leg and a pointer to the BCSM connected to the leg.

Terminating BCSM and Originating BCSM are both objects of the class BCSM in the information model. The SDL model is not exactly constructed this way. Since the O-BCSM and T-BCSM significantly differ, they are modelled in two completely independent process types.

The SDL model of SSF/CCF containes one additional process type: InterfaceHandler. The InterfaceHandler is the permanent manager of the call control function. When the interpretation of the SDL specification begins, the IH is the only process that exists. Then during the course of a call setup the IH, after having received the appropriated messages from the environment, creates the call segment association. It also handles the dialogue eith the SCF and passes the primitives between the Signalling Control Interface and the CSAs and between the SCF interface and the CSAs.

Fig. 3 shows a general example scenarium for IN related to the SDL model. The IBI interface is an internal intra BCSM interface between two half calls.

The full SDL model consits of two half calls as indicated in Fig. 4. Fig. 2 is a refinement of the SSF/CCF block. It shows one half call under control of the SCF, including the service switching and call control functions. In order to operate the model, the Block SSF_CCF needs to be doubled to get two half calls. In this way the full functionality of the interworking between the O-BCSM and T-BCSM can be simulated.

The model has three interfaces. The SigCon interface could, for example, be a DSS1 interface. The Messages on SigCon are abstract and have to be mapped to a real protocol, e.g.DSS1, in a concrete case. IBI is completely internal to a switch. The INAP interface to the SCF is the one with the standardized INAP messages. Because the INAP interface is handeled with the TCAP protocol, a TCAP adapter process is added.

The usage of SDL stands and falls with the availability of powerful tools. In the case of the INAP SDL model, the SDT tool has been used for editing, simulation and validation. Examples of simulation runs with the SDT SIMULATOR are shown in the next section. In addition to the standard SDL tools the SDT-ITEX LINK and AUTOLINK tools [6], [7] have been used to facilitate the generation of the test cases. The test case generation is also explaned in a following chapter.

The SDT VALIDATOR has been used to validate the SDL model in parts according to the method described in [8]. A full and systematic validation proved to be difficult at the beginning because of the size of the model. A large amount of the errors have been found during the test suite development when the SIMULATOR was used. Since the test purposes cover most of the interesting behaviour of INAP, there is some confidence that the model is fairly correct with respect to the intended behaviour. However, a full correctness prove, according to whatever correctness criteria could be defined, was not done and seems to be almost impossible due to the size and complexity of the model.

4 EXAMPLE OF A SIMULATION RUN

The SIMULATOR proved to be useful, in particular in order to debug the model. It is almost impossible to correctly specify a large model like the one for INAP without frequently checking its behaviour by simulation. In particular the fact that a number of different people were involved in the specification process guarantees a lot of specification errors. Even some fundamental errors were found this way.

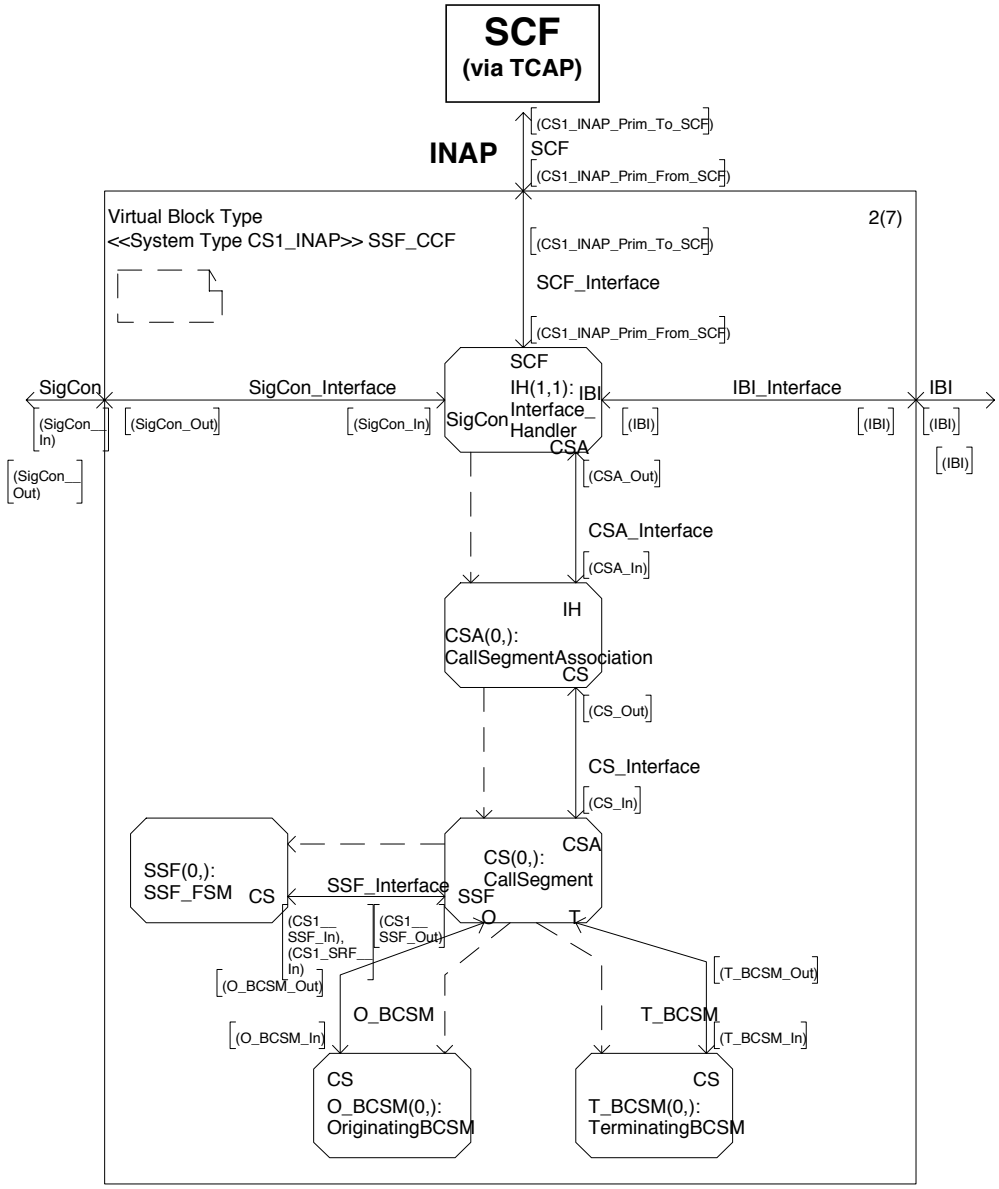


Fig. 2 The INAP CS1 SDL model, half call view

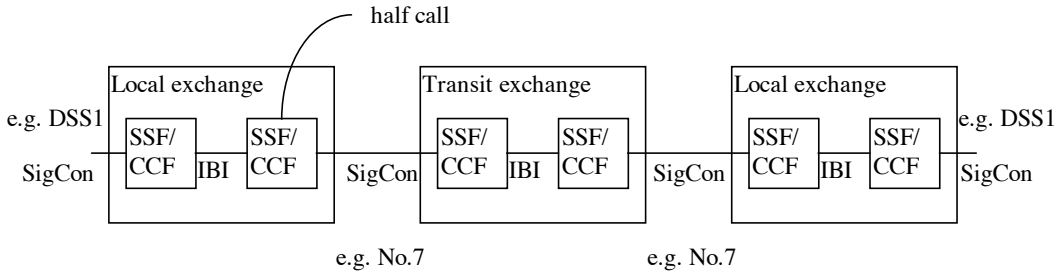


Fig. 3 General example scenarium for INAP showing the SSF/CCF only

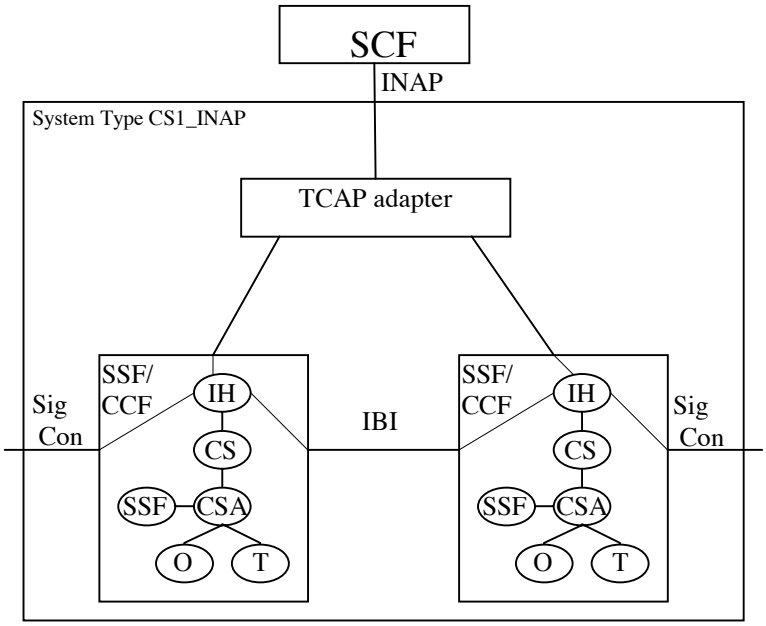


Fig.4 Two half calls

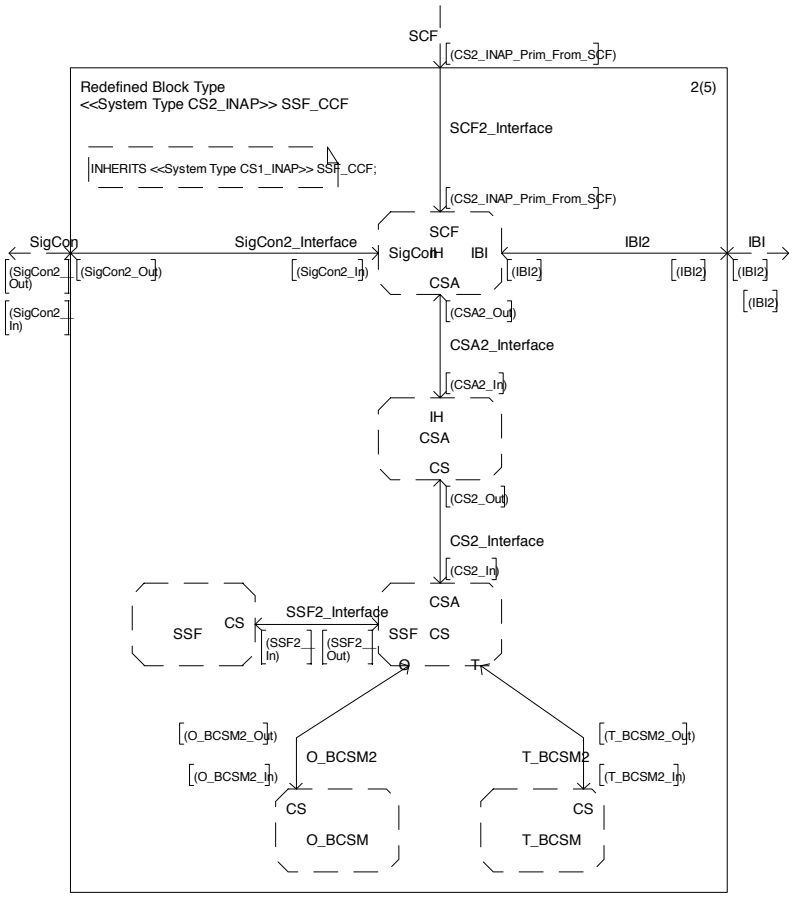
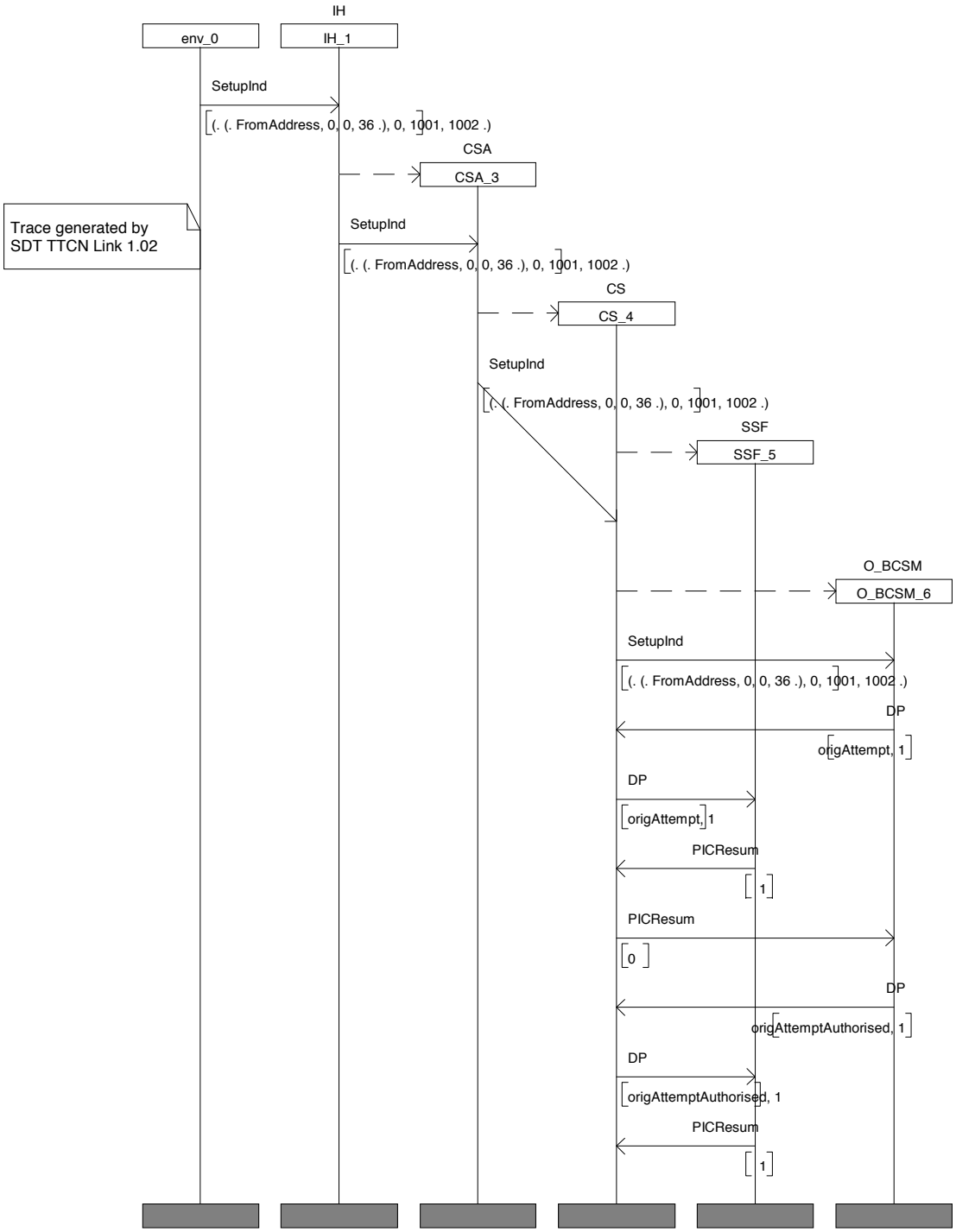


Fig.5 INAP CS2 model which inherits the CS1

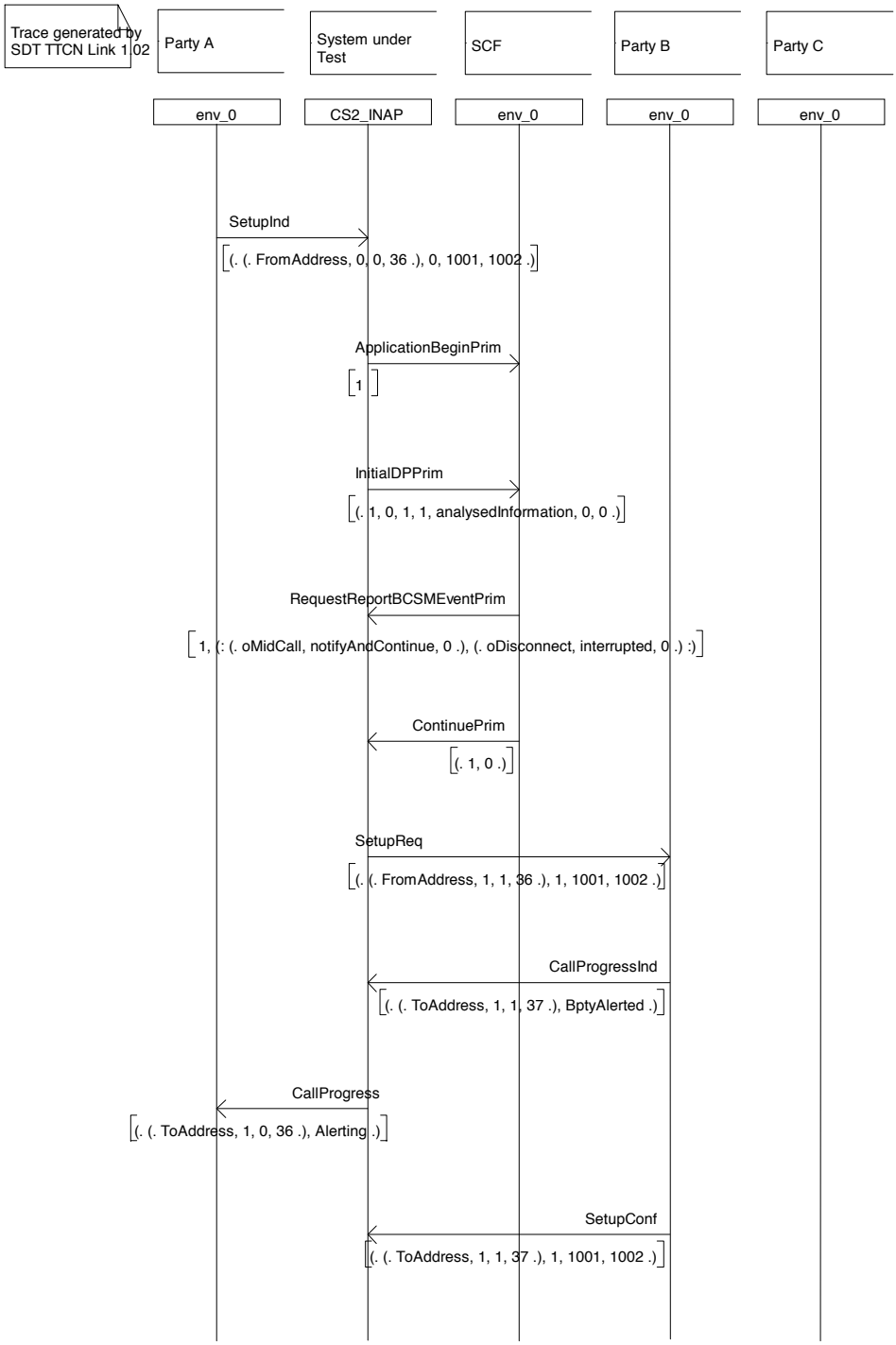
The following message sequence chart (MSC) shows the interworking between the environment (SCF and signalling interface), IH, CSA, CS, SSF and O_BCSM after a SetupInd is received through the SigCon interface. For brevity, only the first few signal exchanges are shown.

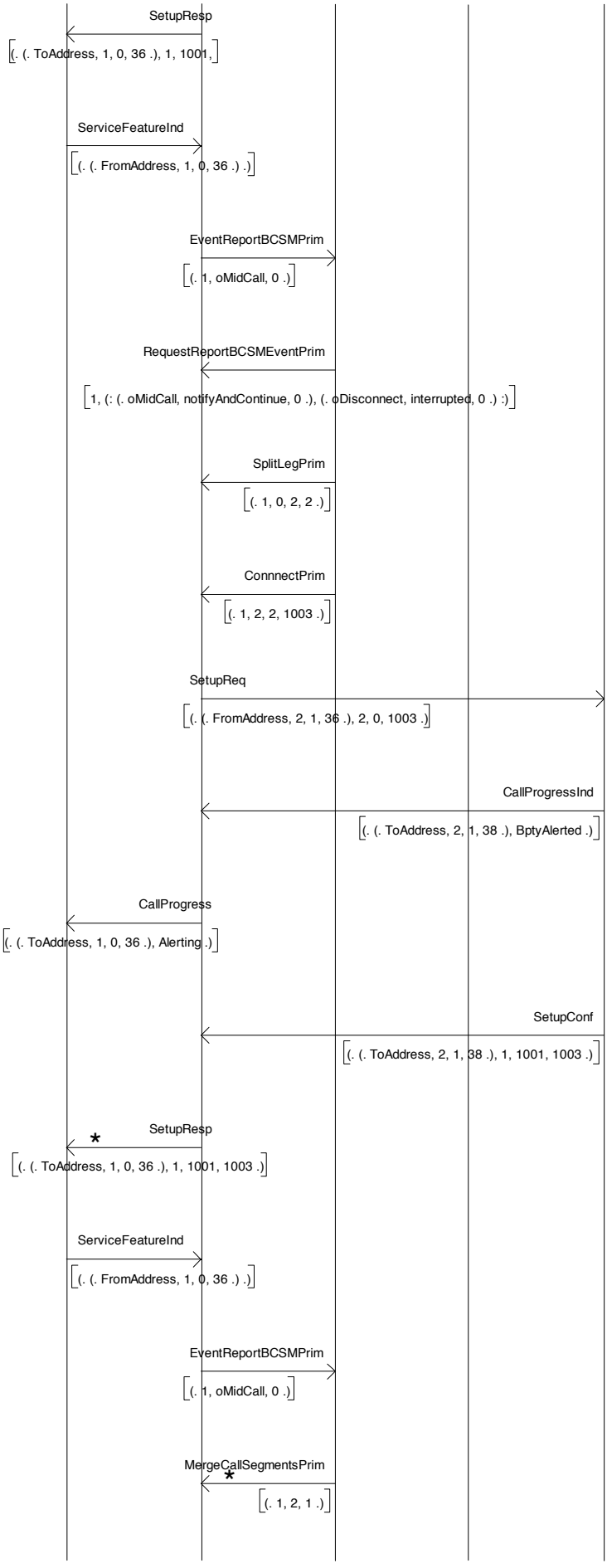


Example for the Three Party Call setup as seen from the environment

The following message sequence chart (MSC) shows an example of the interworking between the SDL model and the environment during a Three Party Call. The MSC is developed according to the architecture of Fig. 4. The internal message exchange of the Two-Half-Call CS2_INAP model is not shown in the MSC. There are three local

exchanges involved: A, B and C. From the point of view of the MSC they are all environments, together with the SCF. For the sake of clarity these environments are each put on a different process instance axis, each one corresponding to one point of control and observation (PCO) as shown in Fig. 6.




SetupResp
[. (. ToAddress, 1, 0, 36 .), 1, 1001,
ServiceFeatureInd
[. (. FromAddress, 1, 0, 36 .) .]
EventReportBCSMPrim
[. 1, oMidCall, 0 .]
RequestReportBCSMEventPrim
[1, (: (. oMidCall, notifyAndContinue, 0 .), (. oDisconnect, interrupted, 0 .) :)]
SplitLegPrim
[. 1, 0, 2, 2 .]
ConnnectPrim
[. 1, 2, 2, 1003 .]
SetupReq
[. (. FromAddress, 2, 1, 36 .), 2, 0, 1003 .]
CallProgressInd
[. (. ToAddress, 2, 1, 38 .), BptyAlerted .]
CallProgress
[. (. ToAddress, 1, 0, 36 .), Alerting .]
SetupConf
[. (. ToAddress, 2, 1, 38 .), 1, 1001, 1003 .]
SetupResp
*
[. (. ToAddress, 1, 0, 36 .), 1, 1001, 1003 .]
ServiceFeatureInd
[. (. FromAddress, 1, 0, 36 .) .]
EventReportBCSMPrim
[. 1, oMidCall, 0 .]
MergeCallSegmentsPrim
*
[. 1, 2, 1 .]

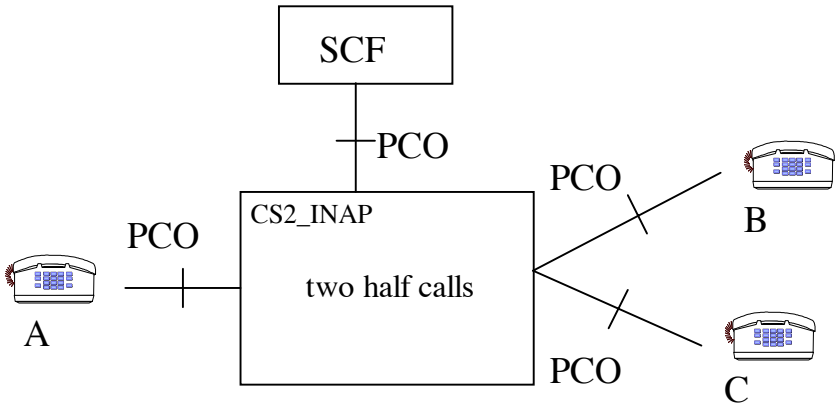


Fig. 6 PCOs for three party call

5 USAGE FOR TEST CASE GENERATION

Service development, feature interaction studies, switch design, test case generation method for TP development, structure of the test suite

The purpose of the SDL model was not only to provide a firm basis for the INAP standard, Annex A of [4], but also facilitate work in different areas, such as service development, feature interaction studies, switch design and test case generation.

ETSI has particular interest in the last point, test case generation. The expectation was that through the use of advanced tools the development of a test suite could be facilitated. The authors were involved in a project team that was set up for developing a test suite.

The development of the test purposes (TP) was done in two steps.

1) A rough TP was defined by hand. It illustrates the basic behaviour in Message Sequence Chart (MSC)-like form[2] which is expected from the implementation under test (IUT). The rough TP does not contain all the constraints in all detail. The rough TP makes reference to a preamble and a postamble.

2) A detailed TP is developed by simulation. The reason for using simulation is that it can be done step by step while the SDL model prompts the developer for the correct items, e.g. PCOs, PDUs, formats, etc. The usual test case is much smaller than the example of above. It consists of reusable preamples and postambles and a test body that tests only a few operations in combination.

An MSC for documentation contains only references to constraints. The constraints themselves are not in the MSC, because they may be very large and confuse the understanding of the basic behaviour. The MSC for documentation is edited on the PR (phrase representation of MSCs) level. Constraints are replaced by identifiers (e.g. IDP_constraint_1, ...). The constraints are then placed outside the MSC.

Pre- & Postambles were developed during the simulation. Autolink does not support the reference mechanism of MSC'96. Pre- & Postamble are therefore normal MSCs. The referencing is done for documentation only for the moment.

Those sequences of events which were likely to be reused frequently were good candidates for preambles and postambles, but individual Pre- and Postambles for a test case are not excluded.

The simulator, when calculating a path through the system, always follows one alternative when more than one is possible. In some cases a range of values is allowed on receive statements. They result from different paths in the system. Since these cases are not very frequent the ranges were added later by hand. The simulation was done only once for one value.

6 CONCLUSIONS

The result of this effort is the INAP SDL model an ETSI test suite for CS2 derived out of it.

The main motivation for the tool-based approach for test case generation was the expected time saving. Test case development is a costly activity. Usually there is only little voluntary support from individual companies, although the test cases themselves are very welcome by a large number of users. From the tool-based approach through the use of SDL, expectations arise about a less expensive test suite production process. Unfortunately it is not easy to obtain exact data which prove a specific time saving. To do this in a particular case, two teams are needed which start at the same level, with the same average skills. One team does the test case generation manually and the other through SDL. Such an experiment is very expensive if it is to be done with a practical example such as INAP. But if it was performed this way, one would have one item of data. In order to achive a statistically relevant statement, one would have to do many such experiments. This has never been done, for obvious financial reasons.

However, one can give a subjective estimate and compare the manpower needed with other similar projects. From this comparison it seems reasonable to state that there is some considerable time saving, if the time and effort spent for making the SDL model is not fully counted. In the case of the INAP protocol the SDLs were done for various reasons, among them for achieving a firm and solid definition of the protocol mechanisms. Therefore the SDL existed irrespectively of the test suite development.

7 REFERENCES

[1] ITU-T Z.100: Specification and Description Language SDL, Geneva, 1996.

[2] ITU-T Z.120: Message Sequence Charts MSC, Geneva, 1996.

[3] ITU-T Q.1218: Intelligent network application protocol, CS1, 1997.

[4] ITU-T Q.1228: Intelligent network application protocol, CS2, 1997.

[5] Ellsberger, J., D. Hogrefe, A. Sarma: SDL - object oriented language for communication systems, Prentice-Hall, 1997.

[6] M. Schmitt, J. Grabowski, D. Hogrefe, B. Koch, A. Ek: Autolink – Putting SDL-based test generation into practice, in: Proceedings of the 11th International IFIP Workshop on Testing of Communicating Systems (IWTCS'98), Tomsk, Aug. 1998.

[7] A. Ek, J. Grabowski, D. Hogrefe, R. Jerome, B. Koch, M. Schmitt: Towards the Industrial Use of Validation Techniques and Automatic Test Generation Methods for SDL Specifications, in: Proceedings of the Eighth SDL Forum, Paris, 1997.

[8] Hogrefe, D.: Validation of SDL Systems, Computer Networks and ISDN Systems, Elsevier Science, vol. 28, no.12, 1996.