SDL- and MSC-based specification and automated test case generation for INAP

Jens Grabowski and Dieter Hogrefe

Institute for Telematics, University of Lübeck,
Ratzeburger Allee 160, D-23538 Lübeck, Germany
Email: {jens,hogrefe}@itm.mu-luebeck.de

Abstract

The development of the Core INAP CS-2 standard and the corresponding conformance test suites by expert teams at the European Telecommunications Standards Institute (ETSI) are historical breakthroughs for the use of SDL and MSC within the international telecommunications standardization process. For the first time, the textual description of a standard has no priority over the corresponding SDL specification. The power of a standard SDL specification has been shown by the successful application of computer aided test generation methods for the production of the necessary standard conformance test suites. This paper introduces the Core INAP CS-2 protocol specification and describes the test generation procedure.

Keywords: SDL, MSC, TTCN, ASN.1, conformance testing, automatic test generation, standardization, ETSI, ITU-T

1 Introduction

With a complete SDL model for the European version of the Intelligent Network Application Protocol (Core INAP), the European Telecommunications Standards Institute (ETSI) is exploring new grounds. Traditionally, the specifications published by ETSI have used SDL [18] only in an informal and illustrative way. This has advantages and disadvantages. The advantages are, e.g., that compared to formal specifications, informal descriptions are more understandable and require less development time. Disadvantages are, e.g., that the specifications are not machine processable and sometimes include ambiguities.

The SDL work for Core INAP at ETSI was done in the Technical (Sub-)Committee SPS3 (TC SPS3) on a voluntary basis with support from the Permanent Expert Group (PEX) at ETSI and the ETSI Technical Committee Methods for Testing and Specification (TC MTS). Unlike other SDL models for INAP, the ETSI model has been done as part of the standardization process and is published together with the standard as a normative (electronic) annex [5].

Reasons for ETSI to develop the Core INAP SDL specification have been the facilitation of service development, feature interaction studies, switch design and test case generation. Traditionally, the development of test suites for conformance tests of standardized protocols has been a major activity of ETSI. Unfortunately, this test suite development has not been successful in some cases. There are some reasons for this: First, test suites have often been developed too late. The products are already on the market before the corresponding test suites are published. A second reason is cost. Because of the risk that the value is limited, the motivation of the companies to participate in the development of a test suite voluntarily is sometimes low. A third reason is quality. Informal descriptions tend to contain ambiguities which may lead to misinterpretations. Even though the approval of a conformance test suite by ETSI requires several reviews at different stages of the test suite development process, the consistency between protocol specification and corresponding test suite cannot always be guaranteed. The development of complete formal SDL descriptions as the normative part of protocol or service specifications is a possibility to tackle these problems.

In most cases, the development of a conformance test suite starts with the identification of the test purposes of the individual test cases. A common representation for test purposes is the Message Sequence Chart (MSC) language [20]. Most commercial SDL tools, like SDT [28] or ObjectGEODE [30] provide possibilities to check whether an MSC diagram[1] describes a behavior included in an SDL description.

The development of the test suite should start in parallel with the SDL modeling. The MSC test purpose descriptions are requirements for the SDL specification and their validation against the SDL description ensures that these requirements

[1] The term *MSC* is used for a diagram written in the MSC language and the language itself. Where necessary, we distinguish between both by using the terms *MSC language* and *MSC diagram*.

are met. In case of changes in the SDL specification, the re-validation of the MSCs can be seen as regression testing, which helps to ensure and improve the quality of the protocol standard. After the finalization of the SDL specification and the MSC test purposes, computer aided test generation (CATG) methods can be applied for the automatic generation of the conformance test suite.

The strength of such a methodology has been tested at ETSI by applying CATG methods to the SDL specification of Core INAP. During test suite development, the validation of the MSC test purposes helped to detect and correct several errors in the protocol specification. Although Core INAP was the first application of CATG methods within ETSI, it was shown that the cost for the test suite development can be reduced significantly.

The remaining parts of this article are organized as follows: Some basics about Intelligent Networks (IN) and the description techniques used for specification are explained in Section 2. The development of the Core INAP SDL specification at ETSI and the SDL specification itself are described in Section 3. The test generation procedure is presented in Section 4. Finally, in Section 5, a summary and an outlook are given.

2 Application area and description techniques

In order to understand the complex working procedures for the development of the Core INAP CS-2 SDL specification and the corresponding conformance test suites, some knowledge of IN, INAP and the languages and notations SDL, MSC, ASN.1 and TTCN is required. It can not be assumed that a reader has expertise in all these areas. Therefore, this section provides an introduction to the most important concepts of IN, SDL, MSC, ASN.1 and TTCN.

2.1 Intelligent Networks and INAP

Intelligent Networks (IN) is currently one of the most important topics in the telecommunications area. IN provides a complete framework for the creation, provision and management of advanced telecommunication services.[2] ITU-T (International Telecommunication Union — Telecommunications Standards Sector) and ETSI standardize IN in several series of standards. These series are known as *capability sets* (CS) and they are distinguished by numbers. Currently, the capability sets 1 (CS-1) and 2 (CS-2) and 3 (CS-3) are published.

Examples of CS-1 services are *abbreviated dialing* (allows the use of short numbers for outgoing calls), *time-dependent routing* (allows incoming calls to be routed based on time, day, week, etc.), *reverse charging* (allows call charges to be allocated to the called party), or *call transfer* (allows a call to be transferred to another destination line).

Instead of adding new services, CS-2 identifies several service categories. The categories refer to *Internet working services*, *call party handling services* (allows to manage various parties' participation within a call), *mobility services*, *broadband services*, *bearer services*, and other service features outside the range of "single ended" calls and/or calls with "single point of control" that were not fully addressed within CS-1.

A main principle of IN is to separate the control of a call and the basic call processing. Conceptually, the control of a call is given to a *Service Control Function* (SCF), whereas the basic call processing is done in *Service Switching Functions* (SSFs). On the implementation side, an SCF is implemented in a *Service Control Point* (SCP) and an SSF is implemented in a *Service Switching Point* (SSP). The SCP is typically a fault-tolerant transaction-processing database that provides call-handling information in response to SSP queries. An SSP is implemented within a normal switch. The following description only refers to the conceptual view, i.e., to SSF and SCF.

Based on call characteristics like call origin or called party number, the SSF detects if a call is an IN call, i.e., the call should be controlled by the SCF. For IN calls, the SSF sends queries to the SCF and asks for information about the handling of the call. Depending on the IN service to be realized for the call, queries have to be sent in different states of the call, and specific call information has to be provided within the queries.

Within the SSF, a call is handled by means of two *Basic Call State Models* (BCSMs) which are called *Originating Basic Call State Model* (O-BCSM) and *Termi-*

[2] Detailed introductions to IN can be found in, e.g., [23] and [29].

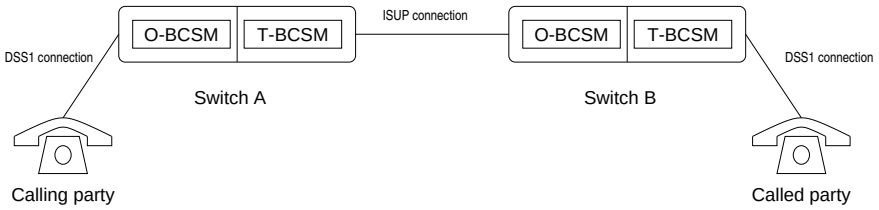

Figure 1: O-BSCM and T-BCSM in SSFs

nating Basic Call State Model (T-BCSM). The O-BCSM describes the incoming side of a call and the T-BCSM models the outgoing side. As shown in Figure 1, the calling party is connected to an O-BCSM and the called party is connected to a T-BCSM.

The BSCMs are finite state machines and are used to control the basic call processing within the SSF. In order to know when to send queries to an SCF, the BCSMs include detection points (DPs). Some DPs have to be armed statically (*trigger detection points*) and some can be armed dynamically (*event detection points*). If an armed DP is reached during the call, the SSF knows that a special treatment of the call is required. In most cases, the SSF has to ask the SCF for further instructions. As a result of such a query, the SCP may provide new information, e.g., a new called party number if a call is transfered to a new destination, or force the SSF to arm a DP.

The communication between an SSP and an SCP is performed by using the *Intelligent Network Application Protocol* (INAP). INAP is defined for different capability sets and for different regions of the world. For example, the term *Core INAP CS-2* refers to the European version of INAP capability set 2 (CS-2).

INAP is normally used within CCS7[3] networks and it is implemented within the *Transaction Capabilities Part* (TCAP) [15] of the CCS7 protocol stack. As shown in Figure 2, INAP is realized on CCS7 level 4 or, with regard to the OSI basic reference model [3], on the application layer (layer 7).

[3]CCS7 is an abbreviation for ITU-T's *Common Channel Signalling System No. 7* (see, e.g., Chapter 10 in [21]).

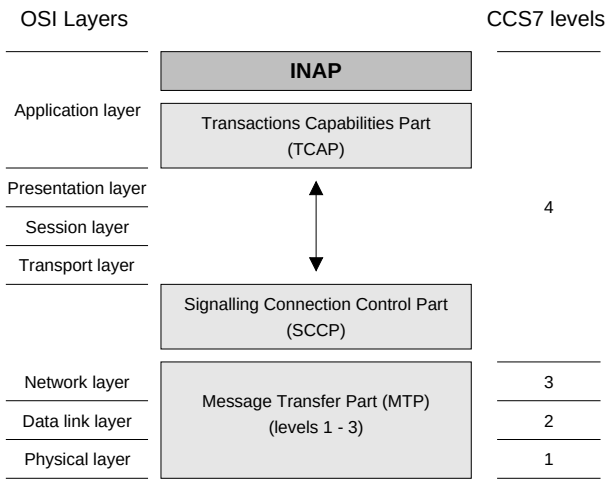

Figure 2: INAP as part of the CCS7 protocol architecture

INAP is defined for the communication between various IN components and not only for the communication between SCF and SSF. However, the Core INAP CS-2 SDL modeling work at ETSI concentrates on the communication between SSF and SCF.

2.2 Description techniques

For the development of Core INAP CS-2 SDL specifications and the corresponding test suites, the description techniques *Specification and Description Language* (SDL) [11, 18], *Message Sequence Chart* (MSC) [25, 20], *Abstract Syntax Notation One* (ASN.1) [27, 16, 17] and *Tree and Tabular Combined Notation* (TTCN) [1, 14] have been used. SDL and MSC are formal description techniques (FDTs), i.e., they have standardized formal syntax and semantics definitions. ASN.1 and TTCN are only notations. They have a standardized formal syntax definition, but the semantics is given informally. ASN.1 and TTCN are well accepted in the telecommunications community for the definition of protocol data and conformance test suites.

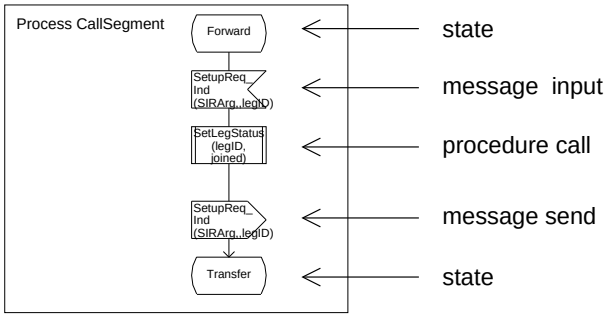

Figure 3: State transition of an SDL process

SDL, MSC and TTCN have two syntactical forms: a pure textual and a graphical representation. The graphical forms are mainly used for editing purposes and documentation. The textual forms are mainly used for the transfer of diagrams and for code generation. Throughout this article, only the graphical representations of SDL, MSC and TTCN are used.

2.2.1 SDL

The formal description technique SDL is standardized by ITU-T as Recommendation Z.100 [18]. SDL is used to specify the behavior of a system. Such a system is a collection of *SDL processes* which communicate asynchronously by exchanging messages. The reception of a message may force a process to change its state. During such a state transition, the SDL process may send new messages and/or perform operations on local variables. Figure 3 presents a state transition of an SDL process specification. If the process CallSegment is in the Forward state and receives the message SetupReq_Ind, it calls the procedure SetLegStatus to perform some operation on a local data structure, sends the message SetupReq and goes into state Transfer.

SDL processes are combined to (sub-)systems by means of *block diagrams*. In a block diagram, the process specifications are referenced and the communication links among the processes and between the processes and the block environment are defined. In Figure 7 the block type SSF_CCF is defined. The six inscribed octagons refer to process definitions and the solid arrows define communication links. The dashed arrows denote dynamic process creation, e.g., in Figure 7 process CS(0,):CallSegment may create process SSF(0,):SSF_FSM. SDL blocks may be combined to bigger blocks or to a complete system. Figure 8 defines the system CS1_INAP. The rectangles refer to block definitions and the solid lines with the attached arrow heads define communication channels.

SDL allows to specify systems in an object-oriented manner. For this, SDL has a type concept for processes, blocks and systems. These types can be reused by means of inheritance and redefinition. SDL types can be collected in SDL packages. The SDL Core INAP CS-2 specification makes extensive use of these object-oriented concepts.

The SDL definition includes many additional language constructs which cannot be introduced here. A complete language description can be found in [2] or [11].

2.2.2 MSC

The MSC language is defined in the ITU-T recommendation Z.120 [20]. Figure 4 shows an example of an MSC. The diagram describes the message flow between the instances SCF, CS2_SSF, SigCon_A and SigCon_B. The instances are represented by vertical axes. The messages are described by horizontal arrows. An arrow origin and the corresponding arrow head denote the sending and consumption of a message. In addition to the message name, parameters may be assigned to messages (see values in square brackets below the message arrows). The send and receive actions along an instance axis are totally ordered. The order of events on different instance axes is mediated by the messages, i.e., a message must be sent before it can be received and consumed.

The rounded rectangles in Figure 4 which cover all instances are *MSC references*. They refer to the MSCs O_OS and ReleaseCallAB. MSC O_OS can be seen as the prehistory of MSC IN2_A_BASIC_RN_CA_01 and MSC ReleaseCallAB as its continuation.

Further constructs of the MSC language describe *instance actions*, *timer handling*, *instance creation*, *instance termination*, the order of events along an instance axis (*coregion*), and the refinement of instance axes by means of *sub-mscs*. Individual

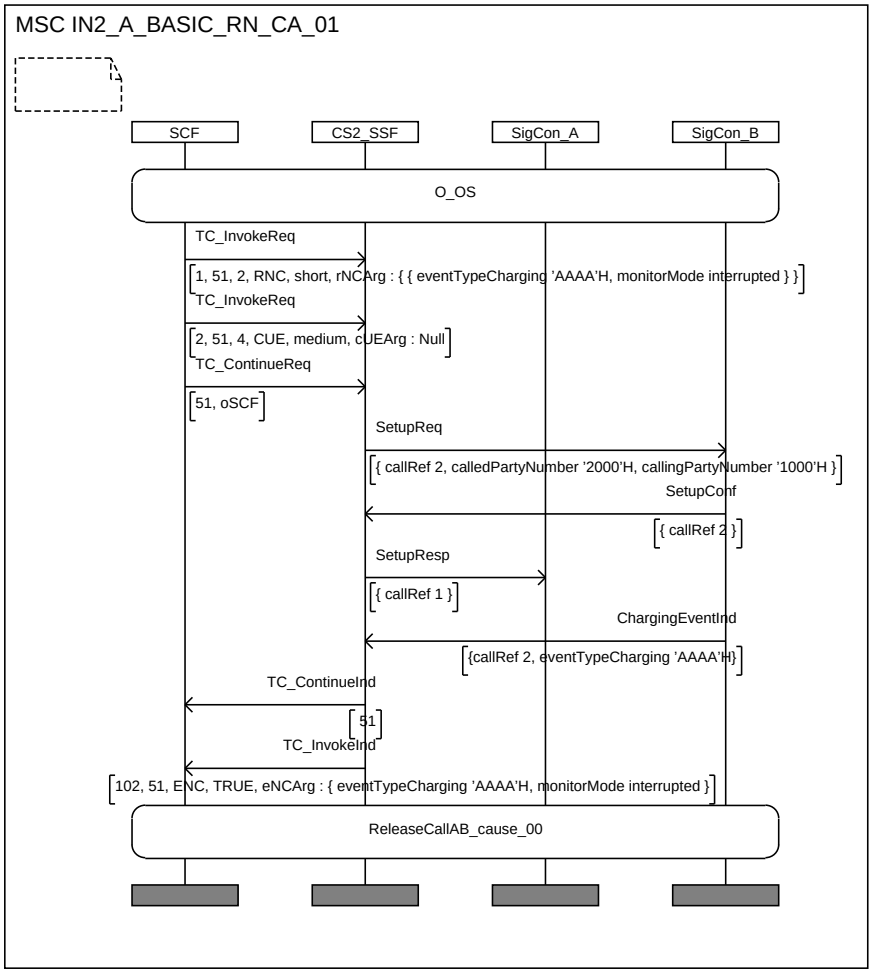

Figure 4: MSC test purpose IN2_A_BASIC_RN_CA_01

MSC sections within one MSC can be combined by means of *inline expressions*. Complete MSCs can be combined by means of *High-level MSCs*. A complete introduction to the MSC language can be found in [25].

2.2.3 ASN.1

The *Abstract Syntax Notation One* (ASN.1) [27, 16, 17] is a notation for the description of structured information intended to be conveyed across some interface or communications medium. ASN.1 allows to specify structured data types and values of the specified types. By means of encoding rules, it is possible to define how data types and values have to be implemented. The close relation to the actual implementation may be one of the reasons why ASN.1 is very popular in industry, but not very well known by scientists.

The use of ASN.1 is supported by TTCN and SDL. For SDL, a special ITU-T recommendation Z.105 [19] exists which defines the use of ASN.1 types and ASN.1 values within SDL. INAP data types and operation calls are specified in ASN.1.

2.2.4 TTCN

The *Tree and Tabular Combined Notation* (TTCN) is defined in Part 3 [14] of the well established *'OSI Conformance Testing Methodology and Framework'* (CTMF), which has been developed and standardized by ISO and ITU-T [13]. TTCN is a notation for the specification of *abstract test suites* for OSI conformance testing. Abstract means that a test suite should be independent from any concrete implementation. A TTCN test suite consists of

- a *test suite overview* which mainly is a table of contents of the test suite,
- a *declarations part* which includes the message and data type definitions,
- a *constraints part* which consists of conditions on message parameters, i.e., default values or value ranges which should be tested, and
- a *dynamic part* which for each test case describes the sequence of exchanged messages.

As indicated by the name *'Tree and Tabular Combined Notation'* (TTCN), a TTCN test suite is a collection of different tables. Figures 5 and 6 present two examples of TTCN tables. They will be explained below.

TTCN has its own data type and value assignment concept. It includes very powerful *matching mechanisms* to express conditions on parameter values. These matching mechanisms are comparable with the wild cards used in UNIX shells. For practical purposes, TTCN allows to use ASN.1 in the declarations and constraints part.

The dynamic part of a TTCN test suite includes the test cases. A TTCN test case describes the sequences of events which should be performed by the testers. In general, these are send and receive events at *Points of Control and Observation* (PCOs). A PCO can be seen as an interface to the *System Under Test* (SUT). The event sequence is specified by means of a tree notation. Figure 5 shows an example. The tree notation can be found in the *Behaviour Description* column.

The tree structure is determined by the ordering and the indentation of the specified events. In general, events with identical indentation denote a branching (i.e., alternative events, for example, lines 8 and 15) and an increased indentation denotes a succeeding event (e.g., lines 2 and 3). Events are characterized by the involved PCOs (i.e., SCF, SigCon_A and SigCon_B), by their kind ("!" denotes a send event and "?" describes a receive event) and by the message which should be sent or received.

The table in Figure 5 includes further information. The entries in the *Constraints Ref.* column refer to TTCN or ASN.1 constraints. An example of an ASN.1 constraint is shown in Figure 6. An entry in the *Verdict* column assigns a *test verdict* to a test run. The verdicts indicate the success of the test run. A *pass* verdict denotes that the test purpose is reached, a *fail* states that an unexpected event has happened and an *inconclusive* describes a situation where neither a *pass* nor a *fail* can be assigned. The example in Figure 5 only includes *pass* verdicts (lines 10, 13, 17). The *fail* cases are specified in the *default behavior description* OtherwiseFail, which is referenced in the test case header.

TTCN allows to structure test case descriptions by using *test steps*. A test step is a behavior tree which can be added to other behavior trees by means of *tree*

Test Case Dynamic Behaviour

Test Case Name : IN2_A_BASIC_RN_CA_01
Group :
Purpose :
Configuration :
Default : OtherwiseFail
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		+O_OS			
2		SCF ! TC_InvokeReq	CIR_RequestNotificationCharging_002(1 , 51)		
3		SCF ! TC_InvokeReq	CIR_Continue_004(2 , 51)		
4		SCF ! TC_ContinueReq	C_TC_ContinueReq_001(51)		
5		SigCon_B ? SetupReq	C_SetupReq({ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H })		
6		SigCon_B ! SetupConf	C_SetupConf({ callRef 2 })		
7		SigCon_B ! ChargingEventInd	C_ChargingEventInd_002		
8		SCF ? TC_ContinueInd	C_TC_ContinueInd_003(51)		
9		SCF ? TC_InvokeInd	CII_EventNotificationCharging_001(102 , 51)		
10		SigCon_A ? SetupResp	C_SetupResp({ callRef 1 })	(PASS)	
11		+ReleaseCallAB			
12		SigCon_A ? SetupResp	C_SetupResp({ callRef 1 })		
13		SCF ? TC_InvokeInd	CII_EventNotificationCharging_001(102 , 51)	(PASS)	
14		+ReleaseCallAB			
15		SigCon_A ? SetupResp	C_SetupResp({ callRef 1 })		
16		SCF ? TC_ContinueInd	C_TC_ContinueInd_003(51)		
17		SCF ? TC_InvokeInd	CII_EventNotificationCharging_001(102 , 51)	(PASS)	
18		+ReleaseCallAB			

Detailed Comments:

Figure 5: Dynamic behavior description of test case IN2_A_BASIC_RN_CA_01

ASN.1 ASP Constraint Declaration	
Constraint Name	: CIR_RequestNotificationCharging_002(Invoke_ID : InvokeIDtype; Dialog_ID : DialogIDtype)
ASP Type	: TC_InvokeReq
Derivation Path	:
Comments	:
Constraint Value	
{ invokeIDtype1 Invoke_ID, dialogIDtype2 Dialog_ID, opClassType3 2, opCodeType4 RNC, timeoutValType5 short, argType6 rNCArg : { { eventTypeCharging PIX_EventTypeCharging1, monitorMode interrupted } } }	
Detailed Comments	:

Figure 6: TTCN constraint CIR_RequestNotificationCharging

attachment. The TTCN test case in Figure 5 includes four tree attachments. In Line 1, the test step O_OS is called and in lines 11, 14 and 18, the test step ReleaseCallAB is attached to the test case behavior.

TTCN supports concurrency by allowing to execute several behavior trees in parallel. For this, a *main test component* (MTC) is allowed to create several *parallel test components* (PTCs). The test components can coordinate themselves during test execution by exchanging *coordination messages*. For the exchange of coordination messages, the same notation as for normal messages is used.

2.3 Tool support

The combined use of SDL, MSC, ASN.1 and TTCN stands and falls with the availability of powerful tools. Within ETSI, the Tau package is used [28]. However, other tool sets provide comparable functionality. As an alternative tool chain, we would like to mention ObjectGEODE for the SDL/MSC side [30] and EXPERT*TTCN [8] for the TTCN side.

Tau contains two tool sets: SDT on the one hand consists of SDL- and MSC-related applications (including support of the combined use of SDL and ASN.1 according to [19]); ITEX on the other hand is used to work with TTCN test suites (including support for the use of ASN.1 within TTCN). Tau provides graphical editors, syntax and semantic checkers, code generators (for several target programming languages), and simulation and validation tools for all of the mentioned description techniques.

2.3.1 Graphical editors and SDL simulator

Graphical editors provide functions to edit and analyze SDL, MSC and TTCN specifications. An SDL specification can be translated into a simulator and a validator application. The SDT simulator provides the possibilities to follow a simulation run by means of an MSC, or by ”highlighting” the SDL symbol which has been executed last in the SDL editor. For further analysis or reuse in another context, a simulation run can be stored in form of an MSC.

2.3.2 The validator

The validator is used to detect dynamic and logical errors in an SDL system. Some of the potential problems are deadlocks, implicit signal consumptions[4] and the sending of signals to non-existing processes.

The Validator is based on state space exploration techniques [12]. The state space of an SDL system is built up in the form of a directed graph, called *reachability graph*. The reachability graph describes the behavior of the SDL system. Its nodes correspond to global system states and its edges represent the transitions between global system states. During validation, the reachability graph is analyzed. For example, a deadlock is found if a node in the graph does not have any outgoing edges.

Verification of an SDL specification against its requirements is one main purpose of the Validator [9]. Most requirements can be expressed in form of MSC diagrams. The Validator explores the state space and searches for a path in the reachability graph complying to the MSC which is checked. The MSC is *verified* if such a path exists.

2.3.3 Autolink

Autolink [10, 22, 26] is part of the SDT validator. The objective of Autolink is to provide an easy-to-use yet powerful tool to generate TTCN test suites from an SDL specification. Potential users are engineers who have a good understanding

[4] SDL processes are allowed to discard signals which are received but not explicitly expected in the actual state.

of the system they have specified, but who do not have detailed knowledge of TTCN. Specialized test suite designers also benefit from using Autolink. They can concentrate on the correct description of test purposes while leaving the error-prone task of writing TTCN code to the tool.

Test generation with state space exploration

Autolink uses the state space exploration techniques and the MSC verification mechanism provided by the SDT validator. The generation of a TTCN test case is based on a *path*. In the Autolink context, a path is defined as a sequence of events which have to be performed in order to go from a start to an end state in the state space of the SDL specification. The externally visible events of a path describe the test sequence to which a TTCN *pass* verdict is assigned.

Autolink uses a modified version of the MSC verification algorithm to compute all relevant transitions in the state space. Each transition is analyzed: Events which are visible at the environment are added to an internal data structure which represents a test case. If an event satisfies the MSC, it is added as a *pass* event; if it violates the MSC, it is added as an *inconclusive* event. Additionally, a constraint is created for every visible event.

After the generation of all test cases, the test suite can be translated into the TTCN format. The declarations part is deduced from the SDL specification; the constraints and dynamic part are a translation of the Autolink internal data structure. The production of the TTCN overview part can be done afterwards by using a TTCN tool like ITEX.

Direct translation of MSCs into TTCN

In order to use a state space exploration to generate test cases from MSCs, a complete SDL specification is required. However in the real (standardization) world, only partial specifications exist for most systems; often there is no SDL specification at all. Standardized protocols like Core INAP CS-2 (Section 3) cannot be specified completely, e.g., error handling or charging functions remain unspecified or are specified partially. Nonetheless, to guarantee a uniform test suite development process, *all* test purposes can be formalized as MSCs.

Autolink supports the processing of manually developed MSC test purposes by providing a function which translates MSCs directly into TTCN (MSC→TTCN translation). Although it does not perform a state space exploration, Autolink performs some static semantics checks.

For this, information information about the interface between the system and its environment is needed. Hence, a minimal SDL specification has to be provided which defines at least the channels to the environment, i.e., the PCOs, and the signals which are sent via these channels.

Constraint handling

Basically, a constraint with a generic name is created automatically for every send and receive event in all test cases. Considering the readability of a test suite, this is far from being optimal. Therefore, Autolink includes a special constraint description language. By defining rules in a configuration file, the test designer can control the naming and parameterization of constraints.

If several test cases are processed consecutively, a lot of constraints are created. Autolink detects and merges identical constraints. As a result, the number of constraints is reduced significantly.

3 The Core INAP CS-2 SDL model

In this section, the working method for the development of the Core INAP CS-2 SDL description and the SDL specification itself are explained.

3.1 Working method

IN is standardized by ITU-T within Study Group 11 (SG 11). The relevant standards are the Recommendations Q.1211-Q.1215, Q.1218-Q.1219 for CS-1 and Q.1221-Q.1225, Q.1228-Q.1229 for CS-2. The ITU-T INAP specifications can be found in the Q.12X8 Recommendations. For the European telecommunications market, the ETSI Technical (Sub-)Committee SPS3 selects an IN subset and adds specific European requirements.

The ETSI work on the Core INAP CS-2 SDL model started in the middle of 1995. It was done in close cooperation with ITU-T SG 11.[5] The goal of the work was to develop a high quality standard which can serve as a basis for validation and test generation, in less than two years.

To reach these goals, it was decided to use ASN.1 as data description language and SDL as specification language for the protocol behavior. Core INAP CS-2 should be developed in an object-oriented manner. As a result, Core INAP CS-1 was developed first and Core INAP CS-2 was modeled on the basis of Core INAP CS-1 by using the SDL mechanisms of inheritance, virtuality and redefinition.

The modeling work was mainly done by a group of voluntary experts from British Telecom, France Telecom, Ericsson, Siemens, Alcatel, Hewlett Packard and Nokia. The group met approximately one week per month at ETSI. The work was supported by an SDL specialist of the ETSI PEX group and resources from ETSI TC MTS.

The modeling work was based on U.S. requirements. As a consequence, a close working relationship with BellCore was set up. The work was structured in such a way that the INAP experts concentrated on the protocol requirements and provided their intentions to the SDL specialists in form of informal SDL. The informal SDL was formalized and the result was discussed and reviewed by the whole group.

Further input on problems and errors was given from the experts group which developed the Core INAP CS-2 conformance test suites. The test suite development by means of CATG techniques started in February 1997, i.e., in parallel to the last phase of the Core INAP CS-2 definition. On the one hand, the work of the test development group lead to changes and corrections of the SDL specification. On the other hand, changes of the SDL specification required some reassessment of the test development group. There is no doubt that the mutual influence of the two groups of specialists[6] helped to improve both the SDL specification and the corresponding TTCN conformance test suites.

The result of the entire modeling work is the SDL Core INAP CS-2 description which consists of more than 450 pages of SDL diagrams. ETSI published Core INAP CS-2 as European Norm (EN) in May 1999. The corresponding ITU-T INAP CS-2 SDL specification was published in Spring 1999. In both standards, the SDL description is a normative annex with the same status as the textual description. As already mentioned by Dave Rayner in [24], the development of the INAP CS-2 SDL description was a breakthrough for the use of SDL in standardization.

[5] Please note, the INAP CS-2 models of ITU-T and ETSI SDL are different, although their development started at the same time with experts contributing to both organizations.

[6] It should be noted that only one expert was member of both groups.

3.2 The SDL specification

Core INAP CS-2 is not a symmetrical protocol. It is used for the communication of different IN components, e.g., SSF, SCF or SRF, with different functions. One would expect different INAP standards for different IN components, but ETSI and ITU-T decided to develop one INAP specification for the SSF only. The reason is simple: In an IN-based network[7], the SSF has to be implemented on all switches, whereas only a few SRF or SCF entities are needed. Therefore, for most telecom operators and manufacturers, the SSF has higher priority than the other components.

As described in Section 2.1 and shown in Figure 1, an SSF handles a call by means of two BCSMs, i.e., the logic of a call is structured into two *half-calls*. Figure 7 presents the half-call structure of the Core INAP CS-1 specification. The process references for originating BCSM (O_BCSM) and terminating BCSM (T_BCSM) can be found at the bottom of the block diagram. They are dynamically created and depending on the role of the half-call, either an O_BCSM or a T_BCSM is created. Figure 7 also includes references to the processes IH of type Interface_Handler, CSA of type CallSegmentAssociation, CS of type CallSegment and SSF of type SSF_FSM.

The IH is a permanent manager of the call control function of the CS-1 half-call view. When the simulation of the SDL specification starts, the IH is the only process of a half-call that exists. During a call setup and after having received the appropriate messages from the half-call environment, the IH creates a CSA. The IH is modeled in such a way that it is able to handle half-calls from several

[7] IN-based means that an IN architecture is used. Some telecommunication services described in IN standards can still be implemented in a conventional environment, i.e., without IN architecture.

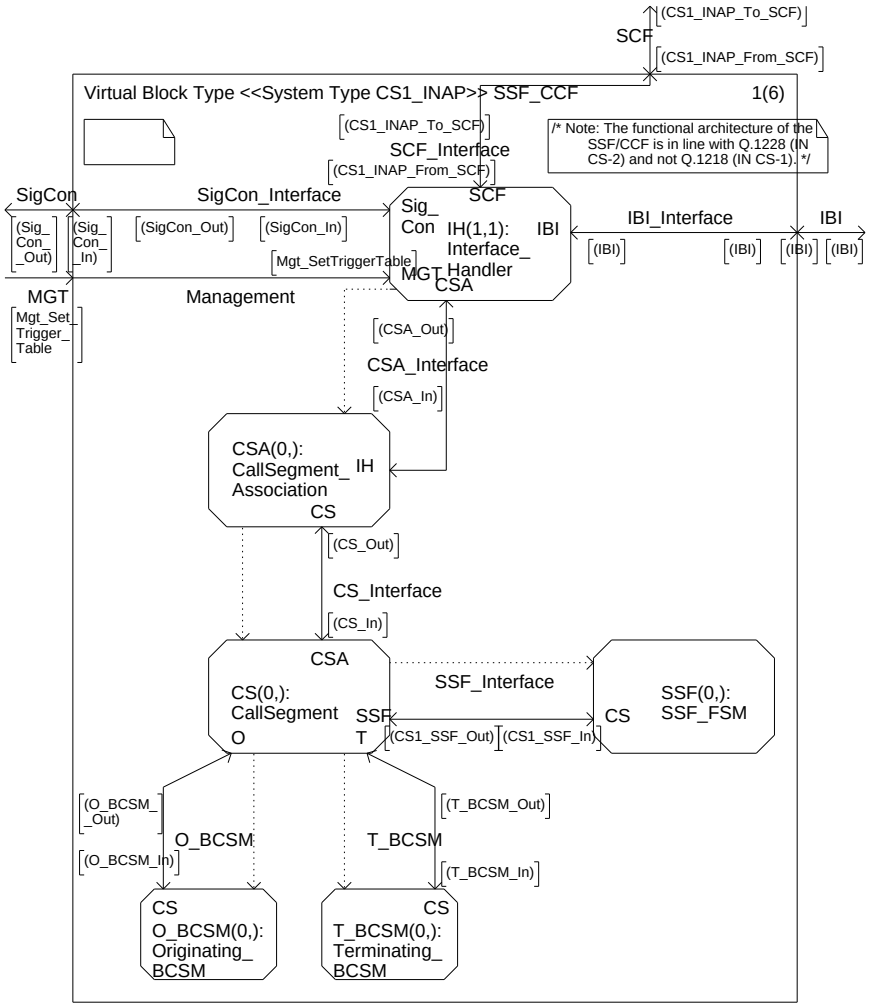

Figure 7: Core INAP CS-1 SSF half-call view

calls. Besides the creation of CSA processes, the IH handles the dialogue with the SCF (via SCF_Interface), manages the dialogue with the other half-call view (via IBI_Interface) and passes messages between the signalling control interface (SigCon_Interface) and the CSAs.

A CSA manages the creation of call segments, i.e., CS processes, and the dialogue with the IH. A CS creates an SSF and an O_BCSM or a T_BCSM. Furthermore, the CS is responsible for the filtering of detection points (see Section 2.1). An SSF process manages the processing of IN operations., i.e, it sets detection points and extracts/stores call information. Furthermore, it is responsible for the handling of detection points, i.e, it controls the arming and disarming of detection points.

For modeling the complete SSF behavior of a switch, two half-call views have to be combined. This is done in Figure 8. The blocks SSF_CCF_A and SSF_CCF_B are instances of the SSF_CCF block type shown in Figure 7. Additionally, Figure 8 includes a third block instance called TCAP_Adapter of type TCAP_Simulator. The reason for this block has been explained in Section 2.1. Within a CCS7 protocol architecture, INAP is normally implemented on top of TCAP. This means that on a standardized interface at the SCF side, INAP primitives are encoded in TCAP messages. In the model this encoding is done by the functionality of the TCAP_Adapter block.

The Core INAP CS-1 system in Figure 8 has five interfaces to the environment: SCF, SigCon_A, SigCon_B, Management_A and Management_B. The exchange of INAP primitives within TCAP messages is performed at interface SCF. The interfaces SigCon_A and SigCon_B are abstract signalling control interfaces. They are used to handle the calls itself. In a real-world implementation, such an interface is connected either to another switch (via SS7) or to a terminal (via DSS1). The interfaces Management_A and Management_B have no counterpart in reality. They are used to set the system into states which cannot be reached by normal message exchange at the other interfaces. They can be compared with some sort of operator terminal at a switch.

Up to here, only the Core INAP CS-1 part of CS-2 has been described. CS-1 services can be characterized by the property that they are applicable to ”single-ended” calls and/or calls with ”single point of control” only. This means that in one SSF, only two half-calls can be involved in a call. Services where more than

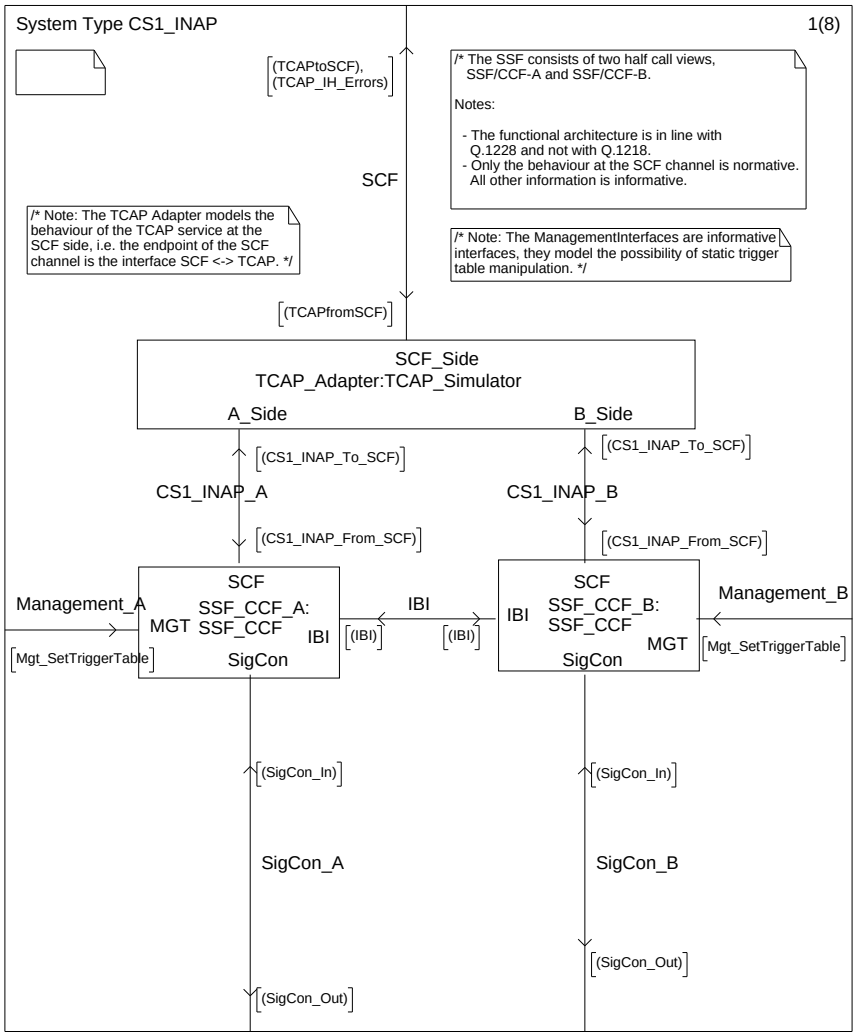

Figure 8: Core INAP CS-1 system diagram

two parties are involved, cannot be realized in CS-1. Therefore, the main extension in the SDL specification of Core INAP CS-2 to CS-1 is *call party handling* (CPH), i.e., the possibility to handle services with more than two parties involved.

For CPH, multiparty calls have to be made visible to the SCF. This is done by introducing the abstract *connection view*. From the perspective of call-related signalling, the connection view is a half-call view. That is, each leg[8] of a multiparty call is associated with a BCSM. In a multiparty call, a leg can have the status *joined*, *pending*, *shared* or *surrogated* and it may have a *controlling* or *passive* role in the call.[9]

In CS-2, *connection view states* for multiparty calls are defined by the legs involved, their status and their roles. CS-2 provides operations to change the state of a multiparty call by using the connection view abstraction, i.e., the operations refer to connection view state changes.

Figure 9 shows the SSF half-call view of the Core INAP CS-2 SDL specification. Compared to Figure 7, the structure does not change much. The CS-1 half-call view is reused and the processes are redefined. In most cases, the redefinitions add behavior to the processes in order to handle the additional CS-2 operations. The connection view handling as described above is performed by the CS process. The CS handles the legs and is responsible for the processing of connection view oriented IN operations.

4 Test generation for Core INAP CS-2

For the understanding of the test generation procedure, it is necessary to have some basic knowledge about the relation between the SDL specification and the test architectures for which the test suites are developed. This is explained in the first part of this section. Then, the test suite development procedure is explained, and finally, the test suites are described.

[8] Legs and half-calls are not exactly identical, but for the understanding of this paper we can assume that they are similar.

[9] A multiparty call can only have one controlling, but several passive legs.

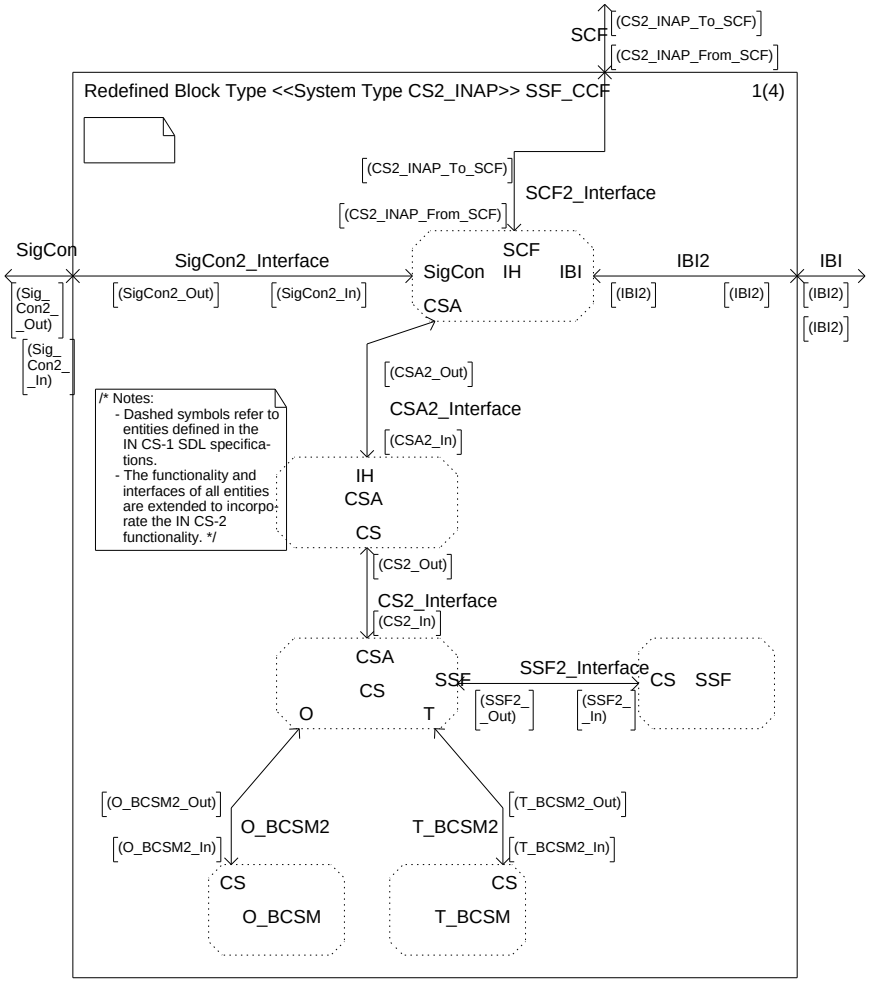

Figure 9: Core INAP CS-2 SSF half-call view

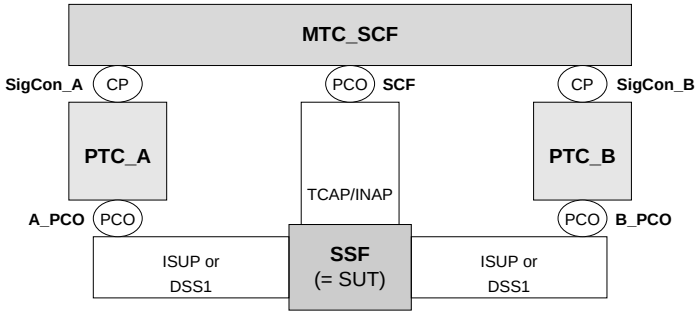

Figure 10: Core INAP SSF test configuration for a two party call

4.1 Multi-party testing context and Core INAP CS-2

The conformance test suites for Core INAP CS-2 are written for a *multi-party context*. The multi-party context is one of the *abstract test methods* defined in CTMF [13]. An abstract test method is an implementation-independent description of a test configuration. Test cases for the multi-party context are specified by using concurrent TTCN (see last paragraph of Section 2.2.4).

Figure 10 shows a Core INAP SSF test configuration for a two party call. There are three test components (MTC_SCF, PTC_A and PTC_B), which control and observe the *System Under Test* (SUT) via standardized interfaces. The TCs describe protocol peer entities of entities which reside within the SSF. The standardized interfaces (SCF, A_PCO and B_PCO in Figure 10) are used as PCOs. They are realized by using standardized communication services. In our case, these services are provided by *TCAP*, *ISUP* and/or *DSS1*.

The Core INAP behavior of the SSF is controlled and observed via TCAP by MTC_SCF at PCO SCF. As indicated by the name, MTC_SCF is the main test component in this test configuration and plays the role of the SCF. PTC_A and PTC_B are parallel test components. They manage the call signalling. Depending on the configuration of the SSF, each of them either plays the role of a terminal or the role of another switch. In case of a switch, ISUP is used for the communication with the SSF; in case of a terminal, DSS1 is used.

As a consequence, four different variants of the same Core INAP test suite are needed. The different variants are the result of the two different roles both, PTC_A and PTC_B, may play.[10]

However, ETSI does not provide different variants of the same Core INAP test suites. Instead, Core INAP abstracts from the concrete ISUP and DSS1 message flow by introducing *abstract signalling control messages* which are exchanged at *abstract signalling control interfaces*. In Figure 8, these abstract signalling control interfaces are represented by the channels SigCon_A and SigCon_B. In a concrete test implementation, the abstract signalling control messages have to be mapped to ISUP or DSS1 messages by the PTCs.[11]

The functionality of the PTCs is reduced to simple mapping functions, since the MTC and PTCs coordinate themselves by exchanging the abstract signalling control messages: A PTC reports the reception of ISUP or DSS1 messages by sending the corresponding abstract signalling control messages to the MTC; the MTC forces a PTC to send ISUP or DSS1 messages by sending the corresponding abstract signalling control messages to the PTC. This means that the SDL system in Figure 8 specifies the mirror behavior of the MTC shown in Figure 10.

The test architecture in Figure 10 describes the situation for two party calls only. For CPH in Core INAP CS-2, multiparty calls have to be handled as well. In the abstract test architecture, further PTCs, CPs and PCOs have been introduced. Similar to the situation above, the test suite for CPH only includes the MTCs of the test cases.

4.2 Test suite development working procedure

The purpose of the SDL model was not only to provide a firm basis for the INAP standard (Annex A of [5]), but also to facilitate work in other areas. ETSI has particular interest in test case generation. The expectation was that through the use of advanced tools, the development of a test suite could be simplified. The tool which has been used for the Core INAP test suite development was Autolink (Section 2.3.3).

A group with experts from Siemens, Alcatel, Telefonica and the University of Lübeck was set up. Additionally, a permanent expert of the ETSI PEX group joined the experts group. The permanent expert was also a member of the SDL modeling group and, therefore, was responsible for the communication between the modeling group and the test experts.

The test suite development for Core INAP CS-2 by means of computer aided test generation methods required knowledge of IN, IN testing, Core INAP, SDL, MSC, TTCN, ASN.1 and CATG tools. None of the experts had deep knowledge in all of these areas. Therefore, during the work sessions, a lot of communication between the experts was required.

The experts team developed three test suites: one for the CS-1 functionality within CS-2, one for CPH and one for SRF. The test suite development procedure was almost identical for all test suites. It was structured into three phases: Identification and development of test purposes, test generation and manual post-processing of the test suite.

4.2.1 Identification and development of test purposes

The development of conformance test suites at ETSI is oriented on test purposes. A test purpose describes a part of the behavior of a protocol for which a test case has to be developed. In a first step, test purposes are specified informally. Afterwards the informal test purposes are formalized by means of MSCs.

Based on the Core INAP CS-2 protocol requirements, the test purposes were identified manually and documented in tables which structure the informal text. As shown in Figure 11, the table entries may refer to pre- and postambles, describe the pass criteria and may provide further information.

Then, MSCs were created for all test purposes. Whenever possible, this was done by simulation of the SDL specification of Core INAP CS-2. An advantage of creating test purpose MSCs by simulation is that the consistency between the in-

[10] Please note, the objective is to test Core INAP and not ISUP or DSS1. The different roles of the PTCs have no influence on the TCAP/Core INAP interface.

[11] To the knowledge of the authors, some telecom operators have defined a mapping of ISUP and DSS1 messages to abstract signalling control messages, but this mapping has been done for internal use only and has not been published officially.

IN2_A_BASIC_RN_CA_01	
Purpose:	Test of **RequestNotificationChargingEvent** base procedure
Requirement ref	
Preamble:	O_OS
Selection Cond.	
Test description	SCF sends **RequestNotificationChargingEvent** invoke to SSF containing mandatory parameters only, with: - ChargingEvent eventTypeCharging, monitorMode (interrupted)
Pass criteria	After triggering of charging event from SigCon_A, check that SSF sends to SCF an **EventNotificationCharging** with the indication of eventTypeCharging
Postamble:	ReleaseCallAB

Figure 11: Informal test purpose description IN2_A_BASIC_RN_CA_01

formally developed test purposes and the protocol is guaranteed. A number of errors in the informal test purpose descriptions were detected with this method.

Since the SDL specification of the Core INAP CS-2 protocol does not include error handling and due to standardization politics, some of the protocol functions are only specified rudimentary. The MSC test purposes related to these protocol aspects were specified manually in order to apply the direct MSC→TTCN translation feature of Autolink. However, these manually generated MSC test purposes look like the ones created by simulation.

The MSC test purposes provided the input for the Autolink tool and were also included in the test purpose document [6]. The inclusion of the MSCs was a requirement from organizations which do not use TTCN for testing, but which need a formal description of each test purpose.

Figure 4 shows an MSC which formalizes the test purpose of Figure 11. The MSC refers to the preamble O_OS and the postamble ReleaseCallAB, which are also described by MSCs.

During the development of the MSC test purposes, the SDL specification of Core INAP CS-2 and the test purposes were validated also. As a result, the SDL specification had to be corrected and modified several times. This changed the behavior of the SDL specification and some of the already developed MSC test purposes became invalid. In order to detect invalid MSCs after each change of the SDL model, all MSC test purposes which had been developed by simulation were revalidated against the SDL model. This was done automatically overnight or at weekends by using a shell script. Due to the complexity of the SDL model, the validation of all MSCs took some time. To reduce it, MSC test purposes were validated in parallel on several computers.

4.2.2 Test generation

Figure 12 presents the test generation procedure from the perspective of the tool. The Core INAP SDL specification was developed by the SDL modeling group. The test purposes (or paths in the Autolink terminology) in form of MSCs had been developed by the test experts group. Additionally, the test experts defined an Autolink configuration file. This configuration file included validator options for test cases which had been generated by a state space exploration and defined the rules for the constraints handling, i.e., naming and parameterization. Based on these inputs the test cases were calculated.

State space exploration was performed by Autolink to generate TTCN test cases for the MSC test purposes created by simulation. The manually specified MSC test purposes were translated directly into TTCN code. Apart from the fact that the test cases related to the manual MSC test purposes do not include event sequences leading to an *inconclusive* verdict, all TTCN test cases look very similar.

4.2.3 Postprocessing of the testsuite

The TTCN output needed postprocessing, because the Autolink version available at that time did not support concurrent TTCN, test suite parameterization by means of *Protocol Implementation Conformance Statement/Protocol Implementation eXtra Information for Testing* (PICS/PIXIT) and timers. The PICS/PIXIT parameterization and the change to concurrent TTCN, i.e., the specification of test configurations, the declaration of CPs and the definition of coordination messages, were performed automatically with a shell script operating on the TTCN file. The timers were introduced manually. The result was analyzed for consistency by using TTCN analyzers and semantics checkers.

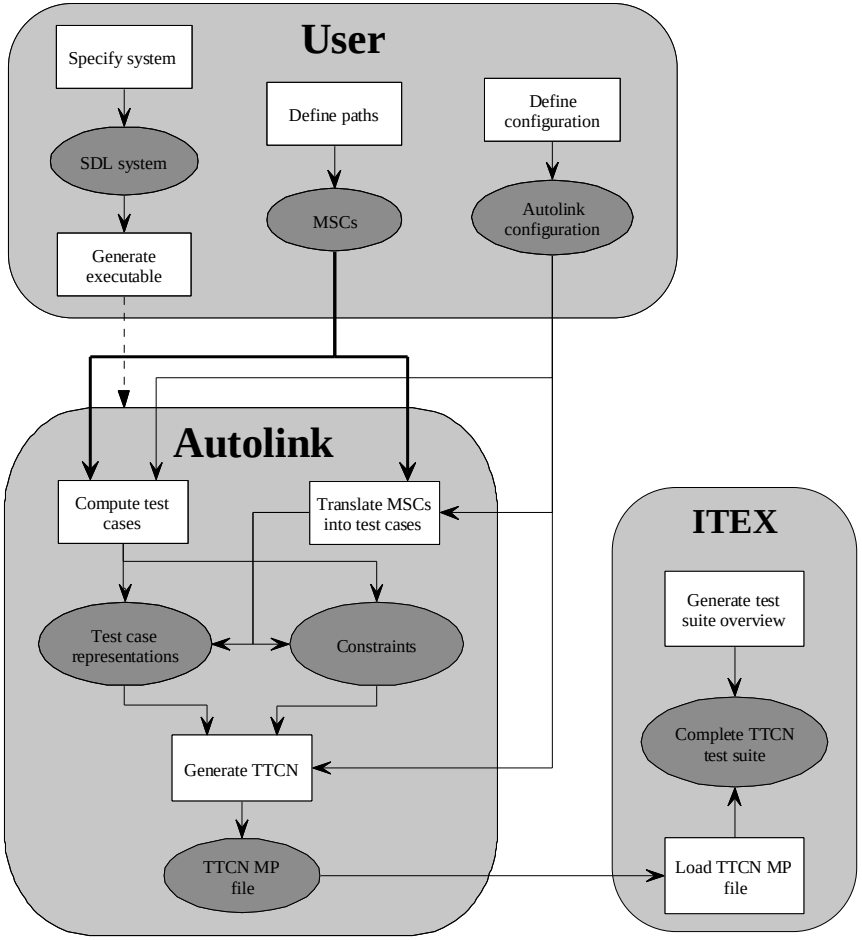

Figure 12: Test generation procedure

A test case after post-processing is shown in Figure 5. This test case is the result of a direct MSC→TTCN translation of the MSC test purpose presented in Figure 4. The corresponding informal test purpose description is shown in Figure 11.

Figure 5 defines the MTC of the test case IN2_A_BASIC_RN_CA_01. The send and receive events in the *Behaviour Description* column of Figure 5 refer to the PCO SCF and the CPs SigCon_A and SigCon_B.

A TTCN expert might be a little bit confused by looking at the MTC description, because the creation of the PTCs is missing. The creation of PTCs is not specified, because the PTCs are not defined in the test suite. For the test case implementation, the consequences of using generic names or omitting the create statements are the same. In both cases, the test case description has to be modified manually.

4.3 The Core INAP CS-2 test suites

Three conformance test suites have been developed for Core INAP CS-2 [7]. They are discussed in this section.

4.3.1 CS-1 functionality within CS-2

The first conformance test suite developed by the ETSI experts group has the objective to test the Core INAP CS-1 functionality within Core INAP CS-2. In total, 126 test purposes were specified [6]. For 67 test purposes, the MSCs could be simulated in order to produce the corresponding test cases by using state space exploration. The remaining 59 MSC test purposes had to be specified manually and translated directly into TTCN due to unspecified parts in the SDL model.

The test suite resulted from a repetitive process of SDL/MSC refinements and modifications, MSC verifications and test generation runs. For statistical purposes, some MSC verification and test generation runs were performed at the University of Lübeck. The test results discussed below were obtained on SUN ULTRA 2 workstations with two 300 MHz processors.

The time needed for the verification of an MSC ranged from 1 min 24 sec to 2 h 15 min. It took between 6 min 44 sec and 51 h 49 min (= 3109 min) to generate a test case.

The larger amount of time needed for test generation is not surprising: During MSC verification, a path in the state space graph is truncated as soon as an event in an SDL transition conflicts with the MSC. During test generation, the path needs to be extended until an observable event occurs.

Verification of all MSCs on a single machine would have taken about a day; generation of all test cases would have taken about a week. Therefore, the processing of test purposes was distributed among up to fifteen workstations.

With regard to the whole development process, the time effort for the actual test generation was not relevant. Most time was spent on refinements of the SDL specification and the test purposes.

For the Core INAP CS-1 test suite, the relation between the number of test cases and manpower spent was one test case per man-day. This includes the development of the test purposes, the set-up of the whole working procedure at the beginning of the test suite development, manual post-processing of the test suite and the production of all documents. In total, ETSI estimates that about 20% of the expenses for the development of the Core INAP CS-1 test suite have been saved by tool support in comparison with manual test suite development.

4.3.2 Core INAP CS-2 CPH test suite

The objective of the second Core INAP test is to test CPH functionality of the SSF. In total, 120 MSC test purposes were defined. 107 MSC test purposes could be simulated and 13 MSC test purposes were specified manually. All test purposes were developed by three experts within two weeks, i.e., 30 man-days. Additionally, 10 man-days were needed for the setup of the test generation, for the post-processing of generated test suite and for the documentation.

However, the relation between the number of test cases and manpower spent was three test cases per man-day. There are several reasons for this impressive result. The working procedure was known and the experts could use their experience from the CS-1 test suite development to optimize their work. Furthermore, the SDL model was much more stable due to the corrections which had been made during the CS-1 test suite development. Only a few errors in the SDL specification were detected and corrected during the development of the CPH test suite.

4.3.3 Core INAP CS-2 SRF test suite

The third test suite checks the INAP connection with an SRF. The test suite consists of 33 test cases. All MSC test purposes were defined manually and the test cases are the result of direct MSC→TTCN translation. The whole test suite including postprocessing and documentation was developed in 20 man-days.

5 Summary and outlook

Core INAP CS-2 is the first protocol in standardization history for which a formal SDL specification has the same normative status as the textual description. Furthermore, Core INAP CS-2 is the first protocol for which the corresponding standard conformance test suites have been developed based on CATG methods for SDL specifications and MSC test purposes.

Core INAP CS-2 is a good example to show that formal description techniques like SDL and MSC are applicable to complex real-world systems, if their smooth interworking with well established techniques like TTCN and ASN.1 is guaranteed. The developed SDL specification is also used outside standardization for the evaluation of service logic, as a tutorial, for the development of in-house tests and as a basis for product design.

The next step in the IN development is CS-3. It was decided that Core INAP CS-3 (ETSI) and INAP CS-3 (ITU-T) should be identical. A Core INAP CS-3 SDL specification is under development at ETSI. The corresponding test suites will also be generated automatically.

Acknowledgements

The authors would like to thank Stefan Heymer, Beat Koch and Michael Schmitt for proofreading and valuable comments on earlier versions of this article.

References

[1] B. Baumgarten, A. Giessler. *OSI conformance testing methodology and TTCN*. Elsevier, 1994.

[2] R. Bræk, O. Haugen. *Engineering Real Time Systems*. Prentice-Hall, 1993.

[3] U. Black. *OSI : A Model for Computer Communications Standards*. Prentice-Hall, 1991.

[4] Cinderella AB. *Cinderella SDL*. `http://www.cinderella.dk`

[5] European Telecommunications Standards Institute. *ETSI Core INAP CS-2; Part 1: Protocol Specification*. Draft European Norm (DEN) 03038-1, ETSI, 1998.

[6] European Telecommunications Standards Institute. *ETSI Core INAP CS-2; Part 3: Test Suite Structure and Test Purposes specification for Service Switching Function (SSF), Specialized Resource Function (SRF) and Service Control Function (SCF)*. Draft European Norm (DEN) 03038-3, ETSI, 1998.

[7] European Telecommunications Standards Institute. *ETSI Core INAP CS-2; Part 4: Abstract Test Suite (ATS) for Service Switching Function (SSF), Specialized Resource Function (SFR) and Service Control Function (SCF)*, Draft European Norm (DEN) 03038-4, ETSI, 1998.

[8] Expert Telecoms. *TTCN*EXPERT product description*. `http://www.expert-telecoms.com`

[9] A. Ek. *Verifying Message Sequence Charts with the SDT Validator*. In: *SDL'93: Using Objects* (editors O. Færgemand, A. Sarma). North-Holland, 1993.

[10] 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: *SDL'97 : Time for Testing – SDL, MSC and Trends* (editors: A. Cavalli, A. Sarma). Elsevier, 1997.

[11] J. Ellsberger, D. Hogrefe, and A. Sarma. *SDL – Formal Object-oriented Language for Communicating Systems*. Prentice-Hall, 1997.

[12] G. J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.

[13] ISO/IEC. *Information Technology — OSI — Conformance Testing Methodology and Framework — Parts 1–7*. ISO, International Standard 9646, 1994 - 1997.

[14] ISO/IEC. *Information Technology — OSI — Conformance Testing Methodology and Framework — Part 3: The Tree and Tabular Combined Notation (TTCN)*. International Standard 9646-3, 1997.

[15] ITU-T Rec. Q.771-775 (1993). *Signalling System No. 7 – Transaction Capabilities*. Geneva, 1993.

[16] ITU-T Rec. X.680-683 (1997). *Information Technology – Open Systems Interconnection – Abstract Syntax Notation ONE (ASN.1)*. Geneva, 1997.

[17] ITU-T Rec. X.690-691 (1997). *Information Technology – ASN.1 encoding Rules*. Geneva, 1997.

[18] ITU-T Rec. Z.100 (1996). *Specification and Description Language* (*SDL*). Geneva, 1996.

[19] ITU-T Rec. Z.105 (1995). *Specification and Description Language (SDL) combined with Abstract Syntax Notation One (ASN.1)*. Geneva, 1995.

[20] ITU-T Rec. Z.120 (1996). *Message Sequence Chart* (*MSC*). Geneva, 1996.

[21] G. C. Kessler, P. Southwick. *ISDN : Concepts, Facilities, and Services - Third Edition*. McGraw-Hill, 1996

[22] B. Koch, J. Grabowski, D. Hogrefe, M. Schmitt. *Autolink – A Tool for Automatic Test Generation from SDL Specifications*. Proceedings of 'Workshop on Industrial Strength Formal Specification Techniques (WIFT'98)', October 21-23, Boca Raton, Florida, 1998.

[23] T. Magedanz, R. Popescu-Zeletin. *Intelligent Networks*. International Thomson Computer Press, 1996.

[24] D. Rayner. *Future Directions for Protocol Testing, Learning Lessions from the Past*. Testing of Communicating Systems, vol. 10, Chapman & Hall, 1997.

[25] E. Rudolph, P. Graubmann, J. Grabowski. *Tutorial on Message Sequence Charts (MSC-96)*. Forte/PSTV'96, Kaiserslautern, October 1996.

[26] M. Schmitt, A. Ek, J. Grabowski, D. Hogrefe, B. Koch. *Autolink – Putting SDL-based Test Generation into Practice*. Testing of Communicating Systems, vol. 11, Kluwer Academic Press, 1998.

[27] D. Steedman. *Abstract Syntax Notation One (ASN.1): The Tutorial and Reference*. Technology Appraisals, 1990, Reprint with corrections 1993.

[28] Telelogic AB. *TAU product description*. `http://www.telelogic.se`

[29] J. Thörner. *Intelligent Networks*. Artech House, 1994.

[30] Verilog SA. *ObjectGEODE*. `http://www.verilog.fr`