26

Applying SAMSTAG to the B-ISDN protocol SSCOP

J. Grabowski, R. Scheurer, Z.R. Dai, and D. Hogrefe
Institute for Telematics, University of Lübeck
Ratzeburger Alle 160, D-23538 Lübeck, Germany, Tel.+49 451 500 37 21,
Fax +49 451 500 37 22, {jens,scheurer,dai,hogrefe}@itm.mu-luebeck.de

Abstract

The test generation method SAMSTAG (SDL and MSC based test case generation) has been applied successfully to the B-ISDN ATM Adaption Layer protocol SSCOP (Service Specific Connection Oriented Protocol). For approximately 70% of the identified test purposes complete TTCN test cases have been generated automatically. In this paper we describe the experiment, discuss the results and explain how further improvements of the test generation process can be achieved.

Keywords

Test case generation, protocol validation, SDL, MSC, TTCN, B-ISDN ATM

1 INTRODUCTION

From 1991 to 1993 Swiss PTT promoted a project at the University of Berne which was aimed at suporting the conformance testing process. One objective was the development and implementation of a method for the automatic generation of abstract test cases in TTCN format based on SDL system specifications and MSC test purposes. As a main result of this project we developed the SAMSTAG method and

Testing of Communicating Systems, M. Kim, S. Kang & K. Hong (Eds)
Published by Chapman & Hall

implemented the SAMSTAG tool [Grabowski, 1994; Nahm, 1994]. The applicability of tool and method has been shown by performing a case study based on the ISDN layer 2 protocol CCITT Rec. Q.921. However, that case study was not complete because we generated test cases for some selected test purposes only, but no complete test suite. The reasons for this incompleteness were restrictions imposed on us by lack of time, money and manpower.

In the years 1993–1995 we improved SAMSTAG by providing mechanisms for dealing with complexity, i.e., the state space explosion problem during test generation [Grabowski et al., 1996]. The most important result of these investigations was the development and implementation of partial order simulation methods for SDL specifications [Toggweiler et al., 1995].

Starting in 1995 we performed another case study based on the B-ISDN protocol SSCOP (ITU-T Rec. Q.2110). The choice of SSCOP was influenced by the interest of the ITU-T in a review of the SSCOP SDL specification and by the need for a test suite for SSCOP. This case study has shown that automatic test generation based on SDL specifications and MSC test purposes is feasible. For 69% of the identified MSC test purposes complete TTCN test cases were generated automatically.

In this paper we focus on describing the application of SAMSTAG to SSCOP. We do not compare SAMSTAG with other methods and tools for automatic test generation. For such a comparison the interested reader may have a look at [Doldi et al., 1996]. The paper proceeds as follows: Section 2 describes SAMSTAG whereas Section 3 introduces the SSCOP protocol. Section 4 explains all steps which have to be performed before the SAMSTAG tool can be applied. The results of the test generation process are presented in Section 5. Section 6 describes the expenses of the test suite development using SAMSTAG. Summary and outlook are given in Section 7.

2 SAMSTAG

SAMSTAG supports the generation of conformance test suites according to IS 9646 [ISO/IEC, 1994]. In this methodology test purposes have to be identified which describe the test case objectives. Test purposes are one basis for test case selection and necessary to relate test results to the protocol functions which have been tested.

The test purposes are implemented in form of abstract test cases by using the TTCN notation. The basis for the implementation is the protocol standard. Currently, this implementation is mainly done manually. SAMSTAG automates this implementation step by using formally specified protocol and test purpose descriptions. As indicated by the abbreviation SAMSTAG which stands for *'Sdl And Msc baSed Test cAse Generation'*, it is assumed that the allowed behaviour of the protocol is defined by an SDL specification, and that the purpose of a test case is provided in form of an MSC.* For the understanding of this paper some basic knowledge of MSC [ITU-TS, 1996b], SDL [ITU-TS, 1996a] and TTCN [ISO/IEC, 1994] is required.

*The SAMSTAG method has been generalised in order to cope with protocol specifications and test purposes which are given in other formalisms than SDL and MSC. The SAMSTAG tool implements the SAMSTAG method for SDL and MSC descriptions.

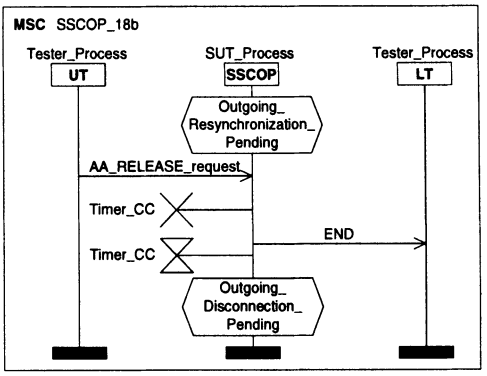

Figure 1 MSC test purpose.

2.1 SAMSTAG input and output

The inputs to the SAMSTAG tool are an SDL specification and an MSC diagram. The test generation process results in a TTCN output.

SDL input
In order to generate the test case SAMSTAG simulates a *closed* SDL system. This means that the SDL specification comprises not only the protocol to be tested, i.e. the *Implementation Under Test* (IUT), but also the tester processes and, optional, other system parts in which the IUT may be embedded or which are required for testing. In the following we use the term *System Under Test* (SUT). An SUT includes no tester processes, but the IUT and all other processes in which the IUT may be embedded or which are necessary for testing.

The specification of a closed system which the IUT only is part of requires additional specification work to be done before test case generation. But, it provides a high degree of flexibility. It allows us to consider different test architectures and to embed the IUT in another system (e.g. [Grabowski et al., 1995]). For simpler cases, tester processes which are able to send and receive all allowed signals at any time can be generated automatically.

MSC input
The SAMSTAG tool accepts test purposes in form of MSCs [ITU-TS, 1996b]. An example is shown in Figure 1. The SAMSTAG tool distinguishes between two types of processes, SUT processes and tester processes. Figure 1 includes one SUT process, SSCOP, and two tester processes, UT and LT. The SSCOP process describes the test purpose from the viewpoint of the SSCOP protocol. In this case, the test purpose is the test of a special state transition. The transition starts in SDL state `Outgoing_Resynchronization_Pending` and ends in state `Outgoing_Disconnection_Pending`. Both states are referred to by means of MSC conditions. During the state transition the SSCOP has to consume an `AA_RELEASE_request` message from UT, cancel timer `Timer_CC`, send `END` to LT and set timer `Timer_CC`

Test Case Dynamic Behaviour

Test Case Name : SSCOP_18b
Group : CONTROL/RESYNC/RELEASE/
Purpose : cf. figures 1 and 6
Default : stddefault

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!AA_ESTABLISH_request	AA_ESTABLISH_request_11F_Y		
2		PRLT?BGN	BGN_111_Y		
3		PRLT!BGAK	BGAK_880_N		
4		PRUT?AA_ESTABLISH_confirm	AA_ESTABLISH_confirm_88_E		
5		PRUT!AA_RESYNC_request	AA_RESYNC_request_35_J		
6		PRLT?RS	RS_352_S		
7		PRUT!AA_RELEASE_request	AA_RELEASE_request_23_G		
8		PRLT?END	END_230_Q		
9		PRLT!END	END_230_Q		
10		PRLT?ENDAK	ENDAK_0_M		
11		PRUT?AA_RELEASE_confirm		PASS	
12		PRLT?POLL	POLL_100_S	INCONC	
13		PRLT?END	END_001_D	INCONC	
14		PRLT?BGN	BGN_111_Y	INCONC	
15		PRUT?MAA_ERROR_indication	MAA_ERROR_indication_P_L	INCONC	
16		PRLT?POLL	POLL_100_S	INCONC	
17		PRLT?END	END_001_D	INCONC	
18		PRLT?BGN	BGN_111_Y	INCONC	
19		PRUT?MAA_ERROR_indication	MAA_ERROR_indication_O_K	INCONC	

Figure 2 TTCN dynamic behaviour description.

again. Hence, to drive SSCOP through this state transition the UT has to send a `AA_RELEASE_request` message and the LT has to receive an `END` message.

TTCN output
SAMSTAG produces complete TTCN test case descriptions including the dynamic behaviour tables, message type definitions, and all constraint declarations. Figure 2 presents the TTCN dynamic behaviour description generated for the MSC test purpose shown in Figure 1. The message exchange related to the test purpose can be found in the lines 7 and 8. The lines 1–6 describe all actions of the tester processes in order to drive the IUT into state `Outgoing_Resynchronization_Pending`, i.e., the state from which the test purpose is observable. The lines 9–11 verify that the test purpose has been performed and drive the IUT back into its initial state. The lines 12–19 are related to inconclusive cases.

2.2 SAMSTAG test generation procedure

For a given SDL specification and a given MSC test purpose the SAMSTAG tool generates a TTCN test case by performing the following steps automatically:

1. The SAMSTAG tool simulates the SDL specification and searches for a trace which (a) starts and ends in the initial state of the SDL specification, and (b) includes the MSC test purpose, i.e., during the trace the MSC is performed. The main problem of step 1 is the state space explosion which may occur during the

search. The SAMSTAG tool provides several mechanisms and techniques to cope with this problem. They are described in Section 2.3.

2. For a test case description only observable events are relevant. Observable events describe actions to be performed by tester processes during a test run. In the following, a trace which includes observable events only is called an *observable*. In step 2 SAMSTAG constructs the observable of the trace obtained in step 1, i.e., all events internal to the SUT are removed. As a result we gain a candidate, called *possible pass observable* (PPO), for a test sequence which may lead to a *pass* verdict. It is only a candidate, because the relation between a complete SDL system trace and its observable is not unique. There may exist other traces which have the same observable, but which do not end in the initial state of the SDL specification, i.e., condition (a) of step 1 is violated, or which do not perform the MSC test purpose, i.e., condition (b) is violated.
3. SAMSTAG tries to verify the uniqueness of the PPO. This is done by contradiction, i.e., by the search for at least one trace which has the PPO as observable, but violates condition (a) or (b) of step 1. If such a trace is found, it is shown that the execution of the PPO during a test run does not ensure that the test ends in the expected state, or does not ensure that the test purpose has been fulfilled. According to IS 9646, in both cases no *pass* verdict should be assigned to such a test sequence. If no such trace is found or if all traces found fulfill the conditions (a) and (b), it is verified that the PPO is a test sequence to which SAMSTAG can assign a *pass* verdict. A verified PPO is called *unique pass observable* (UPO).
4. Due to parallelism, the system may behave in a nondeterministic manner. For testing this means that on a stimulus of a tester process the response from the system may be allowed by the specification, but does not follow the intention of the test case. Neither the test purpose can be verified, nor the specification is violated. According to IS 9646, in such a case an *inconclusive* verdict should be assigned. In order to gain a complete test case description, all traces leading to an *inconclusive* verdict have to be considered. Therefore in step 4 SAMSTAG generates *inconclusive observables* for the UPO found in step 3. An inconclusive observable has prefixes which are identical to prefixes of the UPO, but its last event describes a response from the protocol specification which does not follow the UPO, but is allowed by the SDL specification.
5. Finally, the TTCN test case description for the UPO and the corresponding inconclusive observables are generated, i.e. a TTCN dynamic behaviour description which combines all observables is computed. Additionally, TTCN type definitions and constraints declarations are generated for all messages to be send to and received from the system to be tested. The type definitions are based on SDL signal definitions. The constraints follow from the concrete values observed during the computation of UPO and inconclusive observables. In order to cope with *fail* cases in a final way, a TTCN default behaviour description is generated.

All those five steps above are performed automatically by the SAMSTAG tool. The generated TTCN test cases are complete, i.e., no manual completion is needed. Al-

though the sketched five steps procedure works in many cases, it should be noted that due to theoretical reasons it cannot be guaranteed that PPOs and UPOs exist. The problem of finding PPOs and UPOs can be traced back to the halting problem of Turing machines, for which no solution exists [Hopcroft and Ullmann, 1979].

2.3 Dealing with the state space explosion problem

The main problem of test generation is the explosion of the state space which may occur during the search for the required observables. The reasons for this kind of complexity are (1) the increasing power of modern protocol functions leading to complex specifications, (2) characteristics of the chosen specification language, and (3) missing information about the environment in which the IUT should work.

In our case, characteristics referred to by (2) are related to the interleaving semantics of SDL. The problem of (3) is that in general an IUT is modelled as an open system. For an automatic simulation, during which we search for PPOs and UPOs, the behaviour of the environment has to be modelled. The simple assumption that the environment is able to send and receive any valid signal at any time leads to an enormous amount of possible simulation runs.

Complexity due to (1) cannot be avoided. Therefore, we focus on mechanisms that reduce complexity due to (2) and (3). We distinguish between three classes of reduction mechanisms called *heuristics*, *partial order simulation*, and *optimisation strategies*.

Heuristics are based on assumptions about the behaviour of the system to be tested, or of that of its environment. They avoid the elaboration of system traces which are not in accordance with the selected assumptions. *Partial order simulation* methods avoid complexity which is caused by the interleaving semantics of the specification language. They intend to limit the exploration of traces for concurrent executions*. *Optimisation strategies* intend to reduce the possible behaviour of the system environment. This can be done by using external information, e.g., specifications of surrounding services, or by analysing the specification in order to generate optimal input data to be provided by the tester processes.

In SAMSTAG we implemented several heuristics and partial order simulation methods for SDL specifications. Details can be found in [Grabowski et al., 1996; Toggweiler et al., 1995]. Optimisation has been done by hand. We will come back to this point in Section 4.5.

3 SSCOP

The *Service Specific Connection Oriented Protocol* (SSCOP) is used in the *B-ISDN ATM Adaption Layer* (AAL). The purpose of the AAL is to enhance the services

* A concurrent execution can be seen as a partially ordered set of events. All traces which do not violate the partial order describe the interleaved traces of the concurrent execution.

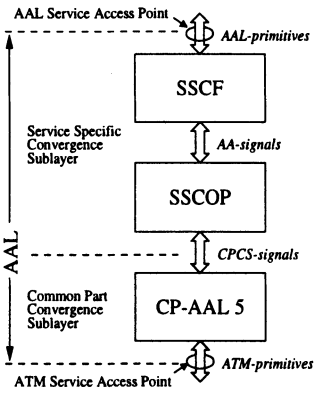

Figure 3 Structure of the ATM Adaption Layer (AAL).

provided by the ATM layer in order to meet the needs of different upper layer applications. One particular AAL type is the *signalling AAL* (SAAL). The SAAL provides communication functions for ATM entities which are responsible for signalling.

As shown in Figure 3, SSCOP can be used within the SAAL. The SAAL is divided into two sublayers, the *Common Part AAL* (CP-AAL) and the *Service Specific Convergence Sublayer* (SSCS). The SSCS comprises an SSCOP entity and a *Service Specific Coordination Function* (SSCF). The objective of SSCF is to map the services provided by the SSCOP protocol to different AAL interfaces. SSCF definitions for *User Network Interface* (UNI) and *Network Node Interface* (NNI) can be found in the ITU-T Recommendations Q.2130 and Q.2140.

3.1 Objective of SSCOP

SSCOP is a connection oriented protocol. Its main purpose is to provide the service of a generic reliable data transfer. In order to implement a reliable data transfer by using the unreliable service of the underlying ATM layer *selective retransmission* is used. This means, all data packets get a sequence number to preserve *sequence integrity*. An SSCOP entity indicates the loss of data packets by sending an `USTAT` PDU. Additionally, SSCOP entities exchange `STAT` PDUs periodically. This is done for keeping track of lost data packets in the special case of lost `USTAT` PDUs. Further characteristics of SSCOP are:

Flow Control. An SSCOP receiver is able to control the rate at which the peer is allowed to send data packets (windowing).

Error Reporting to Layer Management. SSCOP informs the layer management about specific errors such as protocol errors, resynchronization of the connection, or lost data packets.

Keep Connection Alive. SSCOP maintains connections even over periods in which no data transfer is performed. By using a set of timers a connection is partitioned into a *connection control phase*, an *active phase*, a *transient phase*, and an *idle phase*. The

status of a connection is communicated between protocol entities by using POLL and STAT PDUs.

Local Data Retrieval. The SSCOP user is able to retrieve data packets which have not yet been released by the transmitting entity. Different access schemes are provided (full, partial, or selective retrieval).

Protocol Error Detection and Recovery. During operation SSCOP detects errors and triggers a recovery mechanism by exchanging ER and ERAK PDUs with the peer entity.

Connection Control. Connection control is related to establishment, release, and resynchronization of an SSCOP connection. A timer is set to protect against PDU loss during the connection control phase.

3.2 SSCOP SDL specification

The SSCOP recommendation Q.2110 [ITU-TS, 1994] includes an SDL specification which has several informal parts. They refer to system parts and data structures which should not be standardised in Q.2110 or which are defined in another manner, e.g., default values of signal parameter are given in tables. In order to get an executable SDL description as SAMSTAG input all informal parts had to be formalised. In the following the main modifications are listed.

Default parameter and field values to AA-signals and PDUs. Default parameter and field values of SSCOP AA-signals and PDUs are provided in form of tables. These values have to be assigned explicitly before sending. In our SDL specification this is done by inserting extra tasks at appropriate places.

Additional queues and buffers. The SSCOP recommendation introduces additional queues and buffers for dealing with SD, MD and UD PDUs. This is done by means of tasks with informal text which refers to and manipulates these queues and buffers. The informal tasks and the corresponding data structures have been formalised. In a first attempt we implemented it by using SDL, but due to complexity, we changed the implementation language to C++. References and manipulations are made by using the /*#code ... */ construct as used in the SDT tool [TeleLogic AB, 1996].

Priority of internal and external signals. The SSCOP specification distinguishes between internal and external signals. Internal signals are sent and received within SSCOP, while external signals are received from the protocol environment. Internal signals are used to trigger the servicing of the queues and buffers. They are handled by the same message queue that handles external signals. The semantics of SDL would imply that internal and external signals have the same priority. But, in contradiction with the SDL semantics the textual SSCOP description prioritises external over internal signals. When confronted with this problem we decided to follow the SDL semantics and to give internal and external signals the same priority.

Modulo arithmetic. For some state variables which are used for storing counter values or sequence numbers the SSCOP specification introduces modulo arithmetics. We did not model these modulo arithmetics, because SAMSTAG always starts in the

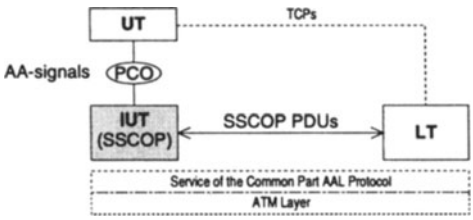

Figure 4 Test method.

initial state of SSCOP and we do not reach the upper bound of affected variables, i.e., modulo arithmetics will never be applied.

3.3 Further modifications

In order to reduce complexity for test generation we implemented some additional modifications and simplifications:

- PDU fields without importance to the function of SSCOP itself have been omitted. All PDUs with variable length have been restricted to a fixed length.
- The handling of PDUs for unassured and management data transfer has been omitted since these features go beyond the scope of conformance testing.
- We abstracted from the CPCS signals by using the PDUs carried by these signals instead.

4 PREPARATORY WORK

Before starting test generation some preparatory work has to be done: (1) a test method and (2) a coverage criterion have to be selected, (3) the structure of the test suite has to be defined, (4) test purposes have to be identified and (5) formalised by means of MSCs, (6) the SSCOP specification has to be adapted to the needs of SAMSTAG , and (7) the tester processes have to be specified. The items (1)-(4) are related to test methodology, the items (5)-(7) are related to SAMSTAG .

4.1 Selection of a test method

IS 9646 [ISO/IEC, 1994] recommends different test methods to be used for protocol conformance testing. These methods mainly differ in the interfaces between tester processes and IUT, and the possibilities to stimulate and observe the IUT during the test. During the definition of our test method we were guided by the distributed test method of IS 9646. Our test method is shown in Figure 4.

There are an upper and a lower interface to the IUT. The upper interface is a point of control and observation (PCO) which is connected to an upper tester (UT). The

UT exchanges AA-signals with the IUT. The lower interface is served by a lower tester (LT). In accordance to IS 9646 we abstracted from the underlaying service, thus the LT exchanges SSCOP PDUs with the IUT. Generally UT and LT coordinate themselves by using *test coordination procedures* (TCPs). We do not model TCPs, because during test case implementation they follow indirectly from the sequence of AA-signals and SSCOP PDUs to be send to and received from the IUT during the test run. Figure 4 does not exactly correspond to the distributed test method as defined in IS 9646. The PCO between IUT and UT is not standardised, i.e., it is not a service access point (SAP). Due to the non-existence of a standardised SAP between IUT and UT it may be more appropriate to use the *remote test method.** In the remote test method there only exist one PCO at the LT interface. Nevertheless, some responses from the IUT have to be triggered by an upper layer user. In the test case descriptions these stimuli are indicated by using the TTCN construct *implicit send event.* Currently, the SAMSTAG tool is not able to deal with the remote test method. For this IUT events which are triggered by upper layers have to be identified and the corresponding *implicit send events* have to be generated.

4.2 Coverage

A test case checks a particular property of the specification. In order to give some confidence that an IUT conforms to its specification, a test suite should cover as much properties of the specification as possible. We based on the SSCOP SDL specification and looked at all state transitions. For each state transition there exists a number of transition control flow paths leading to a next state. They can be seen as properties or test purposes to be tested. Our intention was to generate a test suite that covers all transition paths.

When starting to implement this coverage criterion for SSCOP we discovered two problems: (1) loops which may lead to an infinite number of transition paths with various lengths and (2) the complex state `Data_Transfer_Ready` which due to loops and a cascade of decisions is a starting point for several hundreds of transition paths.

We tackled (1) by setting the maximum number of loop executions during a test run to 1. The problem of (2) was a little bit more complicated. In order to avoid the combination of different decisions we introduced some internal states before decisions and treated them like SDL states, i.e., we split the state transition graph into smaller and less complex pieces.

4.3 The test suite structure

The structure of the SSCOP test suite is shown in Figure 5. It is a tree structure and reflects the SSCOP functionality. The root of the tree represents the whole test suite.

*A detailed discussion on appropriate test methods for ATM AAL conformance testing can be found in [Yoo et al., 1996].

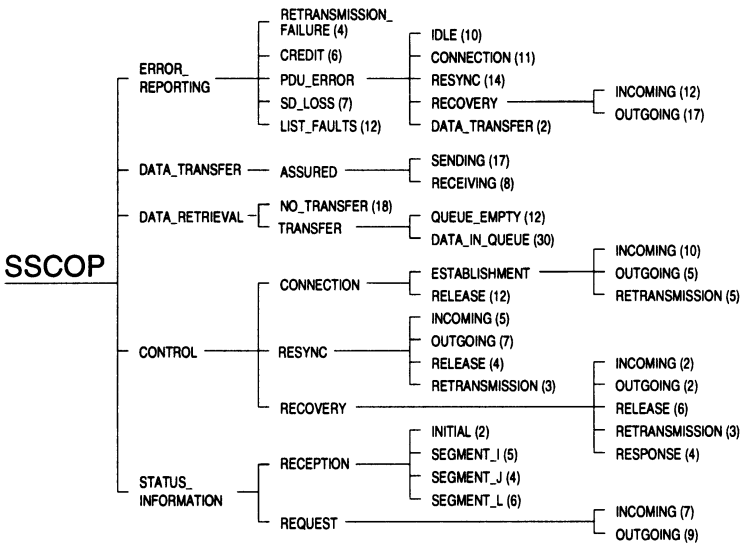

Figure 5 Test suite structure.

Nodes and leafs represent test groups and refer to functions or aspects of SSCOP functions. The test cases in one group should focus on a specific aspect to be tested. The numbers in round brackets following the leaves denote the number of test cases attached to this leaf. The test case SSCOP_18b (Figure 2), for example, is member of the test group called `CONTROL/RESYNC/RELEASE`. The test cases in this group focus on testing the abort of the resynchronization process leading to connection release.

4.4 Identification and specification of test purposes

The identification and specification of the test purposes for the SSCOP test suite follow directly from the coverage criterion (Section 4.2). For each transition path a test purpose is specified. This is done in two steps. In a first step for each test purpose an informal description is produced. In a second step the informal test purposes are formalised by means of MSC diagrams.

An example for an informal description produced for a transition path is shown in Table 1. The informal description is very close to the SDL specification. But, it is aimed to clarify the purpose of a test case and not to specify the entire system behaviour. In case of restrictions imposed by lack of time and money they may be used for the selection of the most important test cases. The formalisation of the test purpose in Table 1 is provided by the MSC in Figure 1.

We identified and specified 281 test purposes. They were assigned to the leaves of the test suite structure according to their functional aspects they focus on. Additionally, in order to do statistical analysis, we arranged the test purposes in 5 groups. This is shown in Table 2. All test purposes of a particular group start in specific states.

Table 1 Informal test purpose description.

Identifier:	SSCOP_18b
Description:	If SSCOP is in state *Outgoing_Resynchronization_Pending* and gets an *AA_RELEASE_request* signal from the SSCOP user, then SSCOP should cancel *Timer_CC*, send an *END* PDU to its peer entity, set *Timer_CC* again, and change into the new state *Outgoing_Disconnection_Pending*.

Table 2 Groups of test purposes.

Group name	*Abbrev.*	*Starting states*	*Number*	*%*
Idle	Idle	Idle	24	9%
Connection Control	ConCo	Incoming_Connection_Pending, Outgoing_Connection_Pending, Outgoing_Disconnection_Pending	51	18%
Resynchronisation	Resyn	Incoming_Resynchronization_Pending, Outgoing_Resynchronization_Pending	38	14%
Recovery	Recov	Incoming_Recovery_Pending, Outgoing_Recovery_Pending, Recovery_Response_Pending	75	27%
Data Transfer	DaTra	Data_Transfer_Ready	93	33%
		total number of test purposes:	281	100 %

4.5 Different models for tester processes

As described in Section 2.1, SAMSTAG needs a closed SDL system as its input. This implies that the tester processes have to be specified as SDL processes.

We started to experiment with general tester processes which are able to send and to receive all allowed signals at any time. But, due to complexity caused by the tester processes we failed even to generate simple test cases. As a result of this experiment we started to use *optimisation strategies* (Section 2.3). This means, we implemented specialised tester processes which stimulate the IUT by using one or more of the following strategies:

- use of special signals to trigger protocol errors;
- use of special sequences of signals as preamble or postamble of the test case in order to reach a particular state quickly;
- focus on a particular signal exchange during which the values of signal parameters are varied;
- specialisation on the role of sender or receiver of data packets.

As a result we gained eight different SSCOP versions which all share the same IUT, i.e., SSCOP process, but differ in the tester processes. Each version focuses on test

Table 3 Number of test purposes covered per version.

	TP Group					*total*	
Version	Idle	ConCo	Resyn	Recov	DaTra	absolute	percental
IDLE	14	31	3	10	0	58	21%
DATA_1	0	0	26	38	16	80	28%
DATA_2	0	0	2	17	4	23	8%
DATA_3	0	0	0	0	3	3	1%
RETRIEVE	10	20	7	10	0	47	17%
RECEIVE	0	0	0	0	17	17	6%
SEND	0	0	0	0	15	15	5%
STAT	0	0	0	0	38	38	14%
total	24	51	38	75	93	281	100%

case generation for one or more selected groups of test purposes. In the following we describe the different SSCOP versions. The Table 3 relates the SSCOP versions to the different groups of test purposes (Section 4.4).

- *IDLE:* This version concentrates on the signal exchange starting in state IDLE and on states dealing with connection control (connecting/disconnecting).
- *DATA_1:* The objective of this version is to cover states dealing with synchronization and recovery of protocol errors.
- *DATA_2 and DATA_3:* These two versions are specialisations of DATA_1. Their objective is to catch the rest of the test purposes that are not directly related to the sending or reception of data packets or the reception of `STAT` PDUs.
- *RETRIEVE:* An important part of the test purposes deal with data retrieval. The data retrieval feature allows the local SSCOP user to retrieve in-sequence data packets which have not yet been released by the SSCOP entity. This is possible in 6 out of the 10 states and requires a preceding phase of data transmission.
- *SEND and RECEIVE:* These two versions concentrate on the sending or reception of data packets.
- *STAT:* This version is very similar to SEND and RECEIVE, but its emphasis is on the transmission of `STAT` PDUs to the IUT. The actual parameter values of the `STAT` PDU are varied.

5 TEST CASE GENERATION

We generated the test suite by using Sun SparcStation 20 and Sun Ultra 2 computers. In this section we describe the result of the test generation procedure and discuss the cases where we failed.

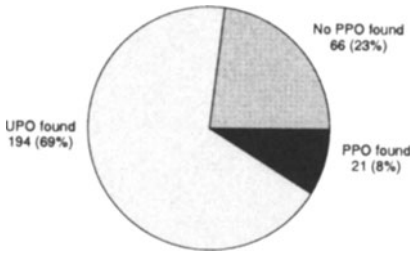

Figure 6 Overall result of the test generation.

5.1 Overall view

For test case generation, SAMSTAG was applied to the different SSCOP models and the 281 test purposes (= 100%). As shown in Figure 6, SAMSTAG generated 194 (= 69%) verified TTCN test cases. For 21 test purposes (= 8%) we found PPOs, but failed to verify their uniqueness (cf. Step 3 in Section 2.2). For 66 test purposes (= 23%) we even did not find PPOs.

In 40% of the cases SAMSTAG generated the verified test cases within 10 minutes, in 53% of the cases it took between 10 minutes and 1 hour. For 7% of the the verified test cases the generation took more than 1 hour.

During the generation process we observed that 3 seconds was the smallest time period to generate a test case and 378 hours was the longest one. For generating the latter one 1 700 560 000 global system states were examined. The longest try where we failed to generate a test case took 837 hours on a Sun Ultra Sparc 2 Workstation. During this attempt more than 2 600 000 000 global system states were investigated.

Our next step was to look at the test generation results for the 5 groups of test purposes. This is done in Figure 7. The groups Idle, ConCo, Resyn, and Recov show approximately the same result. SAMSTAG was able to find UPOs and PPOs for about 80%. The set of PPOs which cannot be verified is relatively small. But, for test purposes related to the DaTra group the result is not that good. The set of found UPOs comprises 46%, the set of PPOs which cannot be verified comprises 17%, and the set of test purposes for which we neither find PPOs nor UPOs comprises 37%. By looking at the SSCOP protocol we see the reason for this result. All test purposes in the DaTra group start in the state `Data_Transfer_Ready` which also is the most complex one. Test generation for test purposes starting in this state is also the most complex part of the entire procedure.

5.2 Failure cases

We identified four reasons for failing to generate test cases:

1. *SSCOP characteristics.* With SSCOP characteristics we refer to the different handling of internal and external signals by the SSCOP protocol. As explained in Section 3.2 the SSCOP standard states that internal and external signals should

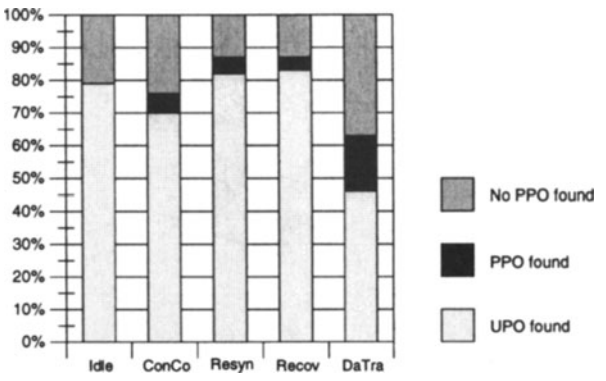

Figure 7 Test generation results related to test purpose groups.

be handled by the same signal queue, but, internal signals should have a lower priority than external ones. This is in contradiction to the SDL semantics. In our SSCOP specification we followed the SDL semantics by assigning the same priority to all signals. As a result we failed to generate test cases for purposes which are somehow related to the different priorities.

2. SAMSTAG *limitations*. The SAMSTAG method is more general than its current implementation. The SAMSTAG tool is a prototype only and includes some restrictions and limitations. In cases where we failed due to SAMSTAG limitations, future SAMSTAG versions may be able to generate test cases.
3. *Complexity*. Due to state space explosion we did not find a trace of the SDL specification which ensures the fulfilment of a given test purpose.
4. *Tester models*. Due to complexity, test case generation using the most general tester process model failed. Furthermore, none of the other models developed turned out to be appropriate to handle these test purposes.

The Figure 8 shows how these reasons are distributed over the failure cases. At least the items 1 and 2, which are responsible for 51 failure cases (= 59%), provide possibilities to improve the result of the test generation process. For item 1 we started discussions with specialists in ITU-T.* In order to align the SSCOP specification to the SDL semantics, the discussions may lead to changes of the SSCOP SDL specification. Failures because of complexity (26 cases, 30%) and inappropriate tester models (10 cases, 11%) need further investigations. At present we cannot say which measures help to avoid these failure cases. Possibly the implementation of further heuristics and optimisation strategies will help.

As shown in Figure 9, the distribution of failures has also been related to the different groups of test purposes. Again the results for the groups Idle, ConCo, Resyn and Recov are comparable. For the test purposes in these groups SAMSTAG mainly fails due to SSCOP characteristics. Complexity and SAMSTAG limitations are only of importance. For the DaTra group we achieved a different result. SAM-

*Within ITU-T, the study group 11 is responsible for the SSCOP recommendation.

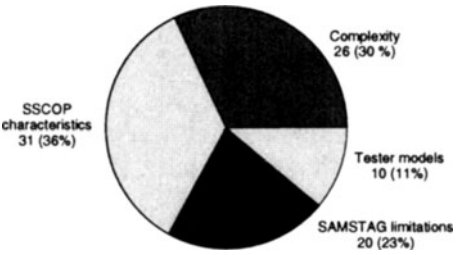

Figure 8 Reasons for failed UPO production.

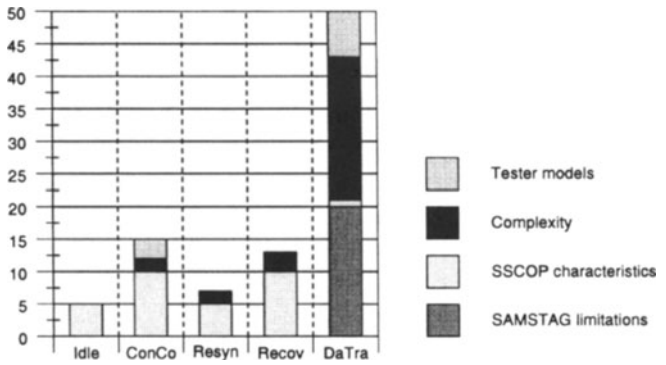

Figure 9 Failure reasons and test purpose groups.

sTAG mainly fails due to complexity and SAMSTAG limitations. Our interpretation is, that this result again reflects the difficulty of generating test cases for the SSCOP state `Data_Transfer_Ready`. We believe that lots of failures due to SSCOP characteristics are hidden in the other failure cases.

6 EXPENSES OF TEST SUITE DEVELOPMENT

The goal of SAMSTAG is to improve the conformance testing process in a twofold manner. On the one hand it should save time and money expenses, and on the other hand the application of SAMSTAG should ensure the consistence between specification and test cases. It is obvious that the latter goal has been achieved. For judging time and money savings a comparison with the expenses for the manual development of such a test suite is required.

For SSCOP such a manually specified test suite [ATM Forum, 1996] exists. The test suite has been developed by the ATM Forum in parallel to our work, but without our knowledge. The main differences to the SAMSTAG test suite are that the ATM Forum test suite is based on the *remote test method* (Section 4.1) and that it has a state oriented structure (Section 4.3). The test purposes identified for both test suites are comparable [Grabowski et al., 1997].

Since there was no data on the ATM Forum test suite available at the time of

Table 4 Development expenses.

Phase	*Subphase*	*Expenses*	*to be performed*
Completion of SDL specification		1 month	manually
Preparatory work	Specification of test method and test suite structure	1 month	manually
	Identification and Specification of test purposes	2 months	manually
	Specification of different tester models	2 months	manually
Test suite generation		1 month	automatically

this writing, we are just able to present our expenses. This is done in Table 4. In the table the development process is structured into the main phases *Completion of SDL specification*, *Preparatory work* and *Test suite generation*. The Preparatory work phase is divided into the subphases which have been described in Section 4. The expenses for test case generation not only include the mere generation time, but also the work of relating test purposes and test models and the experimentation on the application of different SAMSTAG heuristics.

The expenses for test case generation not only include the mere generation time, but also the work of relating test purposes and test models and the experimentation on the application of different SAMSTAG heuristics [Grabowski et al., 1996]. In total the expenses for our case study comprises 7 months. We believe that this is a very good result and that due to increasing experience the expenses for the next protocol will decrease. It should also be noted that SAMSTAG generates test cases for only 70 % of the identified test purposes. We did not estimate the expenses for the manual completion of the test suite. Furthermore we did not estimate the expenses for getting familiar with the SSCOP specification and the SAMSTAG method.

7 SUMMARY AND OUTLOOK

In this paper the application of the SAMSTAG tool to the B-ISDN protocol SSCOP has been described. This case study has shown that automatic test generation based on SDL system specifications and MSC test purposes is feasible. Complete TTCN test cases for 68% of the specified test purposes have been generated automatically. The reasons for cases where SAMSTAG fails to generate test cases have been presented and discussed. For this case study, complexity, SAMSTAG limitations and SSCOP characteristics which violate the SDL semantics are the main reasons for failure. The result of the test generation process may be improved by adding functionality to SAMSTAG and by modifications of the SSCOP SDL specification. Our future work will focus on these aspects and on the extension of SAMSTAG in order to cope with the remote test method as well.

Acknowledgements
The work presented in this paper has been supported partially by the F & E project No. 319, funded by Swiss PTT, and the SPP IF project No. 5003-038997, funded by the Swiss National Science Foundation.

The authors would like to thank all people involved in the development of SAMSTAG. We are also grateful to S. Heymer and the anonymous reviewers providing detailed comments and valuable suggestions which have improved contents and presentation of this paper.

8 REFERENCES

ATM Forum [1996]. Conformance Abstract Test Suite for the SSCOP for UNI 3.1, Technical Committee ATM Forum. af-test-0067.000.

Doldi, L., Encontre, V., Fernandez, J.-C., Jéron, T., Le Briquir, S., Texier, N. and Phalippou, M. [1996]. Assesment of Automatic Generation of Conformance Test Suites in an Industrial Context, *in* B. Baumgarten, H.-J. Burkhardt and A. Giessler (eds), *Testing of Communicating Systems*, Chapman & Hall.

Grabowski, J. [1994]. *Test Case Generation and Test Case Specification with Message Sequence Charts*, PhD thesis, University of Berne, Institute for Informatics and Applied Mathematics.

Grabowski, J., Hogrefe, D., Nussbaumer, I. and Spichiger, A. [1995]. Test Case Specification Based on MSCs and ASN.1, *in* A. S. R. Braek (ed.), *SDL'95 - with MSC in CASE*, Proceedings of 7. SDL Forum, North-Holland.

Grabowski, J., Scheurer, R. and Hogrefe, D. [1997]. Comparison of an Automatically Generated and a Manually Specified Abstract Test Suite for the B-ISDN Protocol SSCOP, *Technical Report A-97-07*, University of Lübeck, Germany.

Grabowski, J., Scheurer, R., Toggweiler, D. and Hogrefe, D. [1996]. Dealing with the Complexity of State Space Exploration Algorithms, *Proceedings of the 6th. GI/ITG technical meeting on 'Formal Description Techniques for Distributed Systems', University of Erlangen.*

Hopcroft, J. E. and Ullmann, J. D. [1979]. *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Company.

ISO/IEC [1994]. Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework, *International multipart standard 9646*, ISO/IEC.

ITU-TS [1994]. ITU-T Recommendation Q.2110: B-ISDN ATM Adaption Layer - Service Specific Connection Oriented Protocol (SSCOP), ITU Telecommunication Standards Sector, Geneva.

ITU-TS [1996a]. ITU-T Recommendation Z.100: Specification and Description Language (SDL, ITU Telecommunication Standards Sector SG 10, Geneva.

ITU-TS [1996b]. ITU-T Recommendation Z.120: Message Sequence Chart (MSC), ITU Telecommunication Standards Sector SG 10, Geneva.

Nahm, R. [1994]. *Conformance Testing Based on Formal Description Techniques and Message Sequence Charts*, PhD thesis, University of Berne, Institute for Informatics and Applied Mathematics.

TeleLogic AB [1996]. *SDT / ITEX 3.0*, TeleLogic AB, Malmö, Sweden.

Toggweiler, D., Grabowski, J. and Hogrefe, D. [1995]. Partial Order Simulation of SDL Specifications, *in* O. Bræk and A. Sarma (eds), *SDL'95 with MSC in CASE*, North-Holland.

Yoo, S., Collica, L. and Jeong, T. [1996]. Conformance testing of ATM Adaption Layer protocol, *in* B. Baumgarten, H.-J. Burkhardt and A. Giessler (eds), *Testing of Communicating Systems*, Chapman & Hall.

9 BIOGRAPHY

Jens Grabowski studied Computer Science at the University of Hamburg, Germany, where he graduated with a diploma degree in 1990. From 1990 to 1995 he was research scientist at the University of Berne, Switzerland, where he received his Ph.D. degree in 1994. Since October 1995 Jens Grabowski is researcher and lecturer at the Institute for Telematics at the University of Lübeck, Germany. His research activities are directed towards network analysis, formal methods in protocol specification and conformance testing.

Rudolf Scheurer received his Diploma degree in informatics from the University of Berne, Switzerland, in 1994. Since his graduation he has been working as a research assistant at the University of Berne and the University of Lübeck, Germany. His research work is focusing on formal methods in protocol specification and conformance testing.

Zhen Ru Dai is an undergraduate at the Institute for Telematics of the University of Lübeck, Germany. She is working as deputy assistant since 1995. From 1995 to 1997 she was involved in the SSCOP case study.

Dieter Hogrefe is a full professor and director of the Institute for Telematics at the University of Lübeck, Germany. His research activities are directed towards formal description techniques and analysis of specifications. He studied Computer Science and Mathematics at the University of Hannover where he graduated in 1983. After a number of years at Siemens Research and the University of Hamburg he was a full professor at the University of Berne and head of the computer networks and distributed systems group from 1989 to 1995. He is an active member of the ITU-T Study Group 10, in which he is the rapporteur for Question 8/10 on verification and testing based on formal methods and is the chairman of ETSI TC MTS.