Combining MSCs and Data Descriptions in order to Generate Executable Test Cases for ISDN Systems

Jens Grabowski[a], Dieter Hogrefe[a], Iwan Nussbaumer[b], Andreas Spichiger[b]

[a]Universität Bern, Institut für Informatik, Länggassstrasse 51,
CH-3012 Bern, ph. +41 31 631 86 81, fax. +41 31 631 39 65

[b]Siemens-Albis AG, Öffentliche Vermittlungssysteme, Steinenschanze 2,
CH-4051 Basel, ph. +41 61 276 71 11, fax. +41 61 276 76 71

ABSTRACT

A method for the automatic implementation of test cases from their specification is presented. For the specification MSCs are used. Special care is taken over the inclusion of message definitions and constraints. For this purpose a new concept for the reference and modification of constraints is introduced. The whole method has been implemented in a set of prototype tools.

Keywords: Protocol Conformance Testing, Test Specification, TTCN, MSC, ASN.1

1 INTRODUCTION

The aim of testing is to detect errors. In the telecommunication area tests are also a prerequisite to ensure the interworking of products from several manufacturers. A typical test environment of our application area is shown in Figure 1. We want to test the functions of the *layer 3 protocol Q.931* [2] within a *Line Trunk Group* (LTG) of an ISDN switching system. The Q.931 protocol is implemented within the LTG and there is no direct access to this implementation. Furthermore, each LTG has only one standardized interface which may be connected to an ISDN end system (e.g. a telephone). The interface of an LTG to the main processor is proprietary.

One possibility to test the Q.931 protocol is to use the whole ISDN switching system as test environment.[1] In this case the test devices access the LTG via standardized interfaces only. The test devices are controlled by a test manager which also records the test results.

The test process of a telecommunication protocol is a procedure which is divided into five phases (cf. Figure 2). The objective of the *analysis phase* is to identify the test cases necessary to check the relevant requirements of the system. The identified test cases

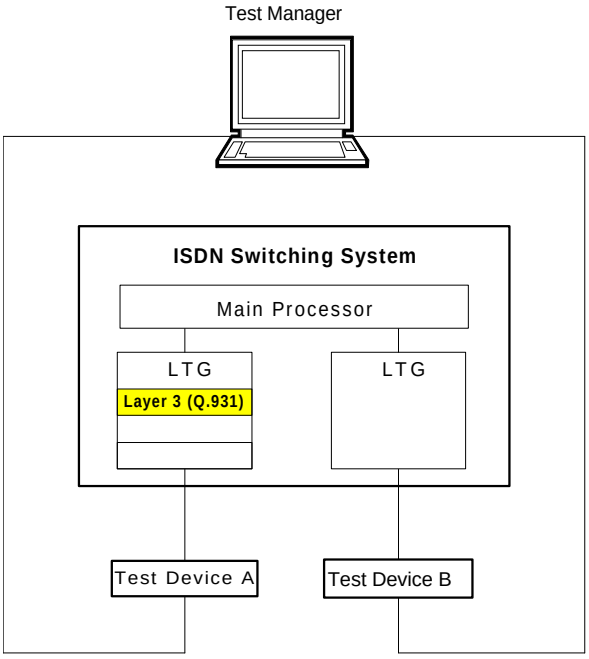

Figure 1: An ISDN test environment

are described in the *specification phase*. During the *implementation phase* the test specifications are transformed into executable programs which drive the test devices and the test manager. In the *execution phase* the test implementations are executed and the test runs are recorded. The test runs are analyzed in the *evaluation phase*. Depending on the result of the analysis the test process may be finished, test runs may have to be repeated, or additional test cases may have to be specified.

Figure 2 describes the ideal test process by showing the entire procedure as a puzzle where all pieces add up. In practice the situation is not so good. The phases are not in accordance with each other and a lot of resources, both manpower and time, are wasted for adapting the results of one phase to the requirements of the next one. Figure 3 indicates this situation, by presenting a puzzle game with pieces which do not

[1]In practice the use of a whole ISDN switching system is very expensive. As a consequence for testing purposes parts of the ISDN system are often only simulated.

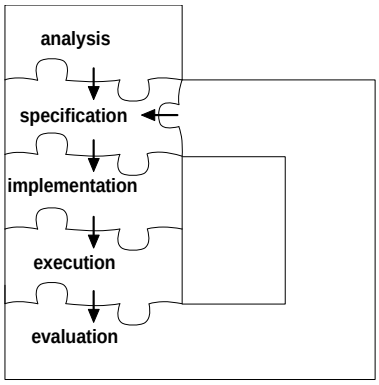

Figure 2: The ideal phase model of the test process

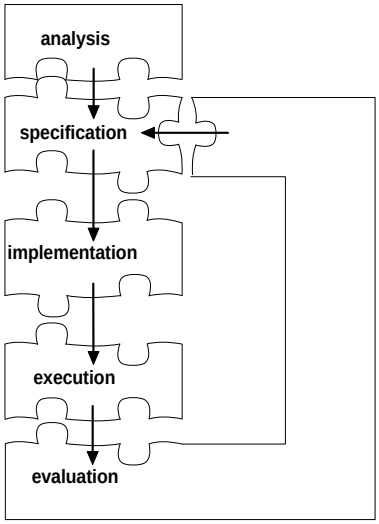

Figure 3: The phase model in practice

match.

Our intention is to improve the test process by providing a smooth transition from the specification to the implementation phase. Subsequently, we present a method, which gives adequate tool support for the specification phase and aims to automate the implementation phase.

2 TEST CASE SPECIFICATION

Figure 4 presents a typical example of a test case specification for the test environment shown in Figure 1. The test case specification is given by informal diagrams and plain text. The shown notation is close to a notation which is used within the Siemens-Albis AG. But, it is not specific to Siemens-Albis. We know from several other telecommunication companies that they use very similar test case descriptions.

Throughout the rest of the paper the test case specification in Figure 4 serves as example. We refer to it by using the test case identifier EDSAOUX.

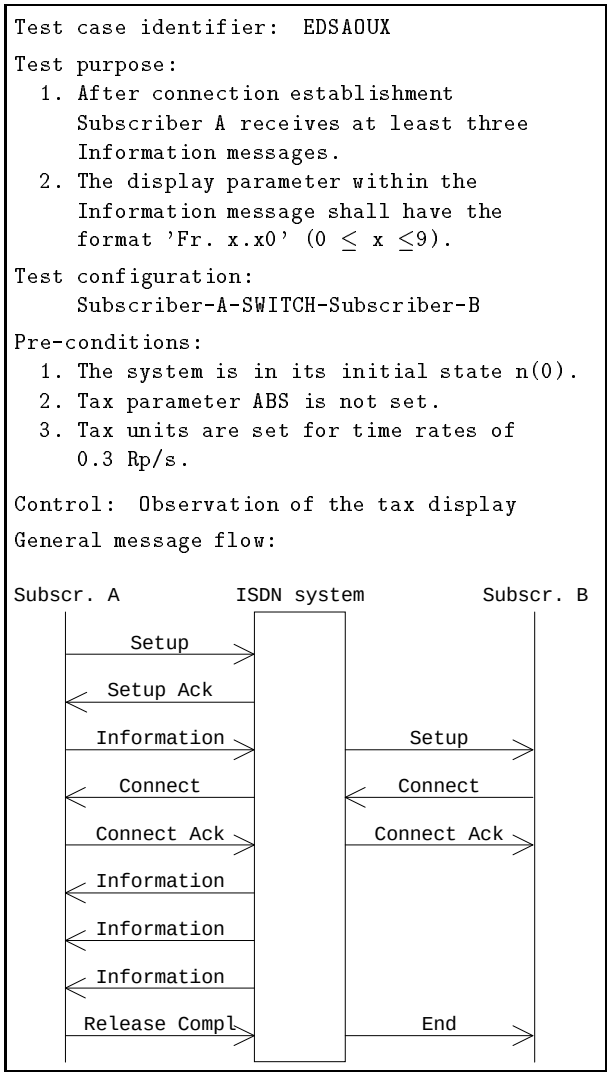

Figure 4: The specification of the test case EDSAOUX

For the presented application example we like to mention that there exist more than 1700 test cases which only test the Swiss specific adaptations of the LTG. They are all specified in the same way as shown in Figure 4 and then implemented by hand. The various possibilities to make errors empazise the need for methods which support the specification and automatic implementation of test cases.

It is obvious that an automatic implementation of test cases cannot be based on informal specifications. Hence, the test specification has to be formalized, but the advantages of the informal specifications should be considered in order to keep the acceptance by the users.

There are two main reasons for the popularity of informal test case descriptions like the one in Figure 4. One is of course the fact that all relevant information of a test case can be written on one page. The second reason is the use of informal diagrams (in our example

it is called *general message flow*) which immediately give an intuitive understanding of the described behavior.

One of the most problematic points of informal test descriptions is that the the data aspect, i.e. structure and values of message parameter, often is hidden in a few informal statements, although data descriptions may provide a lot of implementation work. In our example test case specification EDSAOUX the complexity of message parameter types and values is not visible. The data aspect is tackled in the informal *Test purpose* and the *Pre-conditions* statements.

As a consequence of these facts we looked for a graphical formalism which is almost as easy to use as the shown diagram, but which is formal enough to improve the message flow aspect of the test case implementation (cf. Section 3), and we developed a mechanism which allows to relate data types and data values to messages (cf. Section 4).

3 MESSAGE FLOW DESCRIPTION

We identified the Message Sequence Chart (MSC) language to be adequate for describing the message flow of test cases. MSC is a graphical language, it is standardized by the ITU-T Recommendation Z.120 [4, 8], it has a formal semantics definition [9], and there exist tools which support the use of the language [5, 12, 13].

A small MSC diagram can be found in the upper part of Figure 6. The comparison with the *General message flow* in Figure 4 shows the close relation between both diagram types. Details on the specification of the message flow of test cases can be found in [6].

4 INTEGRATION OF DATA

The MSC language provides no means for describing the data aspects of a message flow. Since data are very important for test cases, we developed an own mechanism in order to relate *data types* and *constraints definitions*, i.e. data values and restrictions on the value range of data types, to messages.

In the following we assume that data types are defined by using ASN.1 and constraints are specified by means of the ASN.1 value notation and TTCN matching mechanisms. ASN.1 and TTCN are standardized [7, 3] and frequently used in the telecommunication area.

Data type definitions

The data types of a protocol are constant for all test cases. They are often provided by the standard documents, or they have to be specified once at the beginning of the test process.

The relations between data type definitions and the messages in an MSC are defined implicitly by the message name. The message name refers to a type definition which itself includes, or refers to the type definitions of the message parameters. We explain this by means of a small example.

The *Information* messages of EDSAOUX (cf. Figure 4) refer to the corresponding ASN.1 type definition *Information_type*. The test case checks a part of the *Display* parameter in the received information messages. The *Display* parameter has the ASN.1 type *Display_type*.

Constraints definitions

Values and restrictions on the value range of message parameters are specified by constraints definitions. We distinguish between two kinds of constraints: *default constraints* and *test case specific constraints*.

For most messages and message parameters the protocol standard provides default constraints. Analogously to types, the relations between default constraints and messages within an MSC are defined implicitly, i.e. for each message type exists a default constraint which assigns values or value range restrictions to the message parameter.

Test case specific constraints are only valid in the context of the test case. They are used

1. to define specific parameter values for messages sent to the implementation under test, and

2. to define test case specific requirements on parameter values for messages received from the implementation under test.

Also the test purpose of EDSAOUX includes a test case specific constraint. It requires to check the format of the *Display* parameter of the *Information* message. The format definition *'Fr. x.x0' ($0 \leq x \leq 9$)* is a constraint on the value range of the *Display* parameter.

Constraints and MSC

There are two possibilities to introduce test case specific value constraints in MSCs. They can be explicitly defined in the MSCs, or they can be defined elsewhere and the MSCs refer to them.

The first possibility is problematic, because the constraints may become too big for the MSC. For example, the constraints of the *Information* of the test case example in Figure 4 comprise two pages. One would loose transparency if the message flow and all constraints are defined in the same diagram.

The second possibility is problematic, because the principle of locality is violated. A reference mechanism may lead to situations where the relevant parts

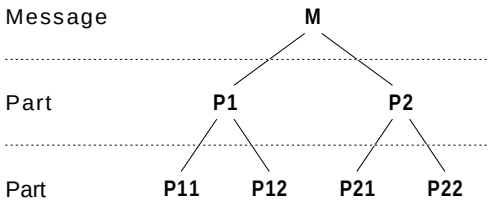

Figure 5: Message structure

of a test case description are defined at different locations. In EDSAOUX the test case specific constraint for the *Information* message would have to be referred to in the test case specification. The test purpose relevant constraint on the *Display* parameter itself would only be referred to in the constraint of the *Information* message. There would be no direct indication of the test purpose in the test case specification.

As a consequence of this we decided to develop a comfortable reference mechanism which allows to refer to self written test case specific constraints, and which provides possibilities to define test case specific constraints by modifying existing constraints. Additionally, it allows to define test case specific constraints directly within an MSC, e.g. if a test case specific constraint only comprises one concrete value.

A comfortable reference mechanism

Our reference mechanism is a reference language, in the following called RL. Within an MSC the statements of RL are related to messages. They can be found in parentheses close to the corresponding message name, or message arrow (cf. Figure 6). This is no extension of the MSC language, because the MSC standard [8] proposes to use expressions in round brackets to assign parameter information to messages.

Schematically, an RL statement has the following shape:

$$(M; P1, P2; P11, P12; P21, P22)$$

It consists of several *parts* which are separated by semicolons. Each part may consist of several *subparts* which are separated by commas. The structure of an RL statement reflects the structure of a corresponding message. A message has a hierarchical structure. A part of an RL statement represents a hierarchy level. The subparts describe elements within a hierarchy level. The message structure of the schematic RL statement presented above is shown in Figure 5. The details on RL can be found in [10].

An example may provide an impression of how the reference mechanism works. The MSC in Figure 6 describes the most relevant part of the message flow of EDSAOUX, i.e. the reception of the three *Information* messages. The inscription of each *Information* message *(; Display:DisplayEDSAOUX)* is an RL statement. It specifies that the constraint for this message

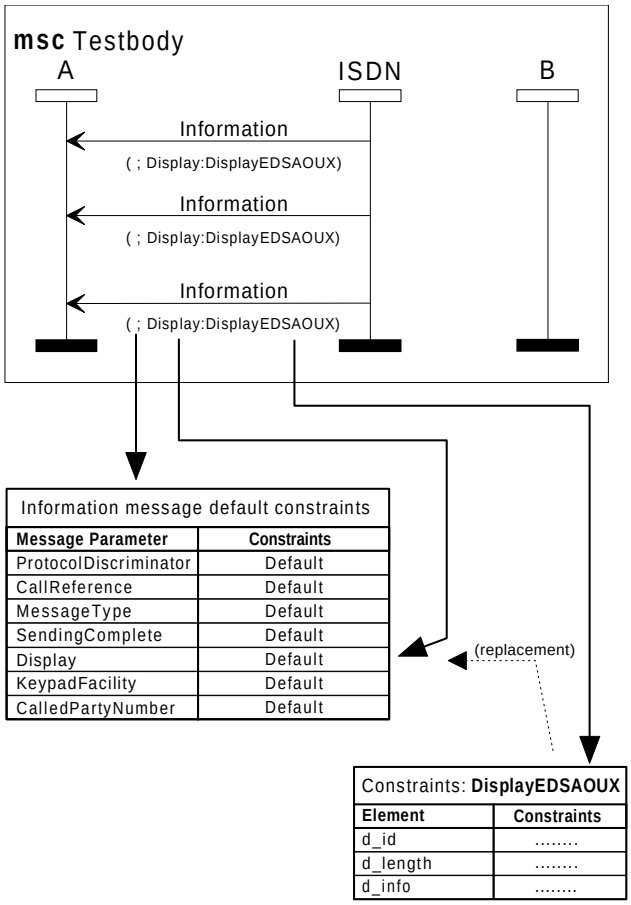

Figure 6: Referring to test case specific constraints

is a modification of the default constraint. The default constraint for the *Information* message shall be used, but the default constraint for the *Display* parameter shall be replaced by the test case specific constraint *DisplayEDSAOUX*.

5 TEST CASE IMPLEMENTATION

As test case implementation language TTCN [7] has been chosen. TTCN supports the use of ASN.1 definitions and there exist TTCN Compilers [1, 11] which allow to generate executable test cases from TTCN descriptions.

As described in the previous sections, we specify test cases by means of MSC, RL, and ASN.1/TTCN data type and constraints definitions. The implementation of such test case specifications can be fully automated. Details on this step can be found in [12, 10].

6 TOOL SUPPORT

We implemented our test case specification method by a set of prototype tools. The core of the tool set

is a graphical *MSC editor* which allows to specify test cases by means of MSCs and RL statements. The *MSC/TTCN generator* transforms MSCs into TTCN message flow descriptions, i.e. the dynamic part of a TTCN test suite. The *TTCN builder* combines the output of the MSC/TTCN generator, and the TTCN/ASN.1 data type and constraints definitions to complete TTCN test cases. The tool set is implemented on a PC in a Windows 3.1 environment.

7 SUMMARY AND OUTLOOK

A method for the comfortable specification and automatic implementation of test cases has been presented. It is close to existing and well established procedures. The choice of the standardized languages MSC, TTCN and ASN.1 allows to use commercial tools for test case specification and test execution. The method is implemented by a set of prototype tools.

For the application of our method in an industrial environment the interface to the reference mechanism for test case specific constraints has to be improved. A statement of our reference language RL statement can be considered to be the minimum information to generate the references to test case specific constraints and to define new constraints which are based on existing ones. But, complicated message structures may lead to complex statements of the reference language RL and without knowledge of the message structure the RL statements may not always be easy to read. The reference mechanism should have no influence on the test case specification process. In near future we will extend the MSC editor by a graphical interface for message constraints. The user will be enabled to check, define and modify constraints without knowledge of the underlying reference mechanism.

ACKNOWLEDGEMENTS

The elaboration of this paper is funded partially by the KWF-Project No. 2555.1 *'Graphical Methods in the Test Process'*. We would like to thank Dr. E. Rudolph for proofreading and F. Bosatta, Ch. Rüfenacht, Dr. R. Schönberger, S. Suter and Ch. Zehnder for their valuable comments and suggestions.

References

[1] Alcatel Network Systems. GSM - Testing by Alcatel. Product Information, Alcatel STR AG (Zürich) and Alcatel SEL AG (Stuttgart), 1993.

[2] CCITT. Recommendations Q.930- Q.940: Digital Subscriber Signalling Systen No. 1 (DSS 1), Network Layer, User-Network Management. The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, 1989.

[3] CCITT. Recommendations X.208 (ISO/IEC IS 8824 & 8824/AD1): Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1) and Addenum 1: ASN.1 Extensions. The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, November 1989.

[4] J. Grabowski, P. Graubmann, E. Rudolph. The Standardization of Message Sequence Charts. In *Proceedings of the IEEE Software Engineering Standards Symposium 1993*, September 1993.

[5] J. Grabowski, D. Hogrefe, R. Nahm. Test Case Generation with Test Purpose Specification by MSCs. In O. Faergemand and A. Sarma, editors, *SDL'93 - Using Objects*. North-Holland, October 1993.

[6] J. Grabowski, D. Hogrefe, I. Nussbaumer, and A. Spichiger. Improving the Quality of Test Suites for Conformance Tests by using Message Sequence Charts. In *Proceedings of the 'Fourth European Conference on Software Quality' in Basel (Switzerland)*, October 1994.

[7] ISO/IEC JTC 1/SC21. Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation. International Standard 9646-3, ISO/IEC, 1992.

[8] ITU Telecommunication Standards Sector SG 10. ITU-T Recommendation Z.120: Message Sequence Chart (MSC). ITU, Geneva, June 1992.

[9] S. Mauw and M. A. Reniers. An Algebraic Semantics of Basic Message Sequence Charts. *The Computer Journal (Special Issue on Process Algebra)*, 36(5), 1993.

[10] Ch. Rüfenacht. Extending MSCs with Data Information in order to Specify Test Cases. Diploma Thesis (written in German), University of Berne, Institute for Informatics, February 1994.

[11] Siemens AG. Product Information K1197, K1103. Siemens AG Berlin, 1993.

[12] S. Suter. The MSC Based Generation of the Dynamic Part of TTCN Test Cases. Diploma Thesis (written in German), University of Berne, Institute for Informatics, January 1994.

[13] TeleLOGIC Malmö AB, Box 4128, S-203 12 Malmö (Sweden). *SDT 2.3*, 1993.