Some Implications of MSC, SDL and TTCN Time Extensions for Computer-Aided Test Generation

Dieter Hogrefe, Beat Koch, and Helmut Neukirchen

Institute for Telematics, University of Lübeck
Ratzeburger Allee 160, D-23538 Lübeck, Germany*
{hogrefe,bkoch,neukirchen}@itm.mu-luebeck.de

Abstract. The purpose of this paper is to describe how computer-aided test generation methods can benefit from the time features and extensions to MSC, SDL and TTCN which are either already available or currently under study in the EC Interval project. The implications for currently available test generation tools are shown and proposals for their improvement are made. The transformation of MSC-2000 time concepts into TTCN-3 code is described in detail.

1 Introduction

Computer-aided test generation (CATG) from system specifications has been an active field of research for many years [1,5,10,24]. This research has resulted in the development of a number of test generation tools [2,8,9]. Today, two industrial-strength, commercially available CATG applications exist [6,18]. These tools take formal system specifications with the 1992 edition of the *Specification and Description Language (SDL-92)* and test purpose descriptions with the 1996 version of *Message Sequence Charts (MSC-96)* as input and produce test suites based on the second edition of the *Tree and Tabular Combined Notation (TTCN-2)* [14].

Meanwhile, the standards of both SDL and MSC have been updated (SDL-2000 [16], MSC-2000 [15]) and a thoroughly new version of TTCN has been standardized (TTCN-3 [7]). In addition, the European Commission has set up the *Interval* project [21] to prototype an SDL, MSC and TTCN-based tool chain for the development and testing of systems with real-time constraints. During the first project stage, the Interval consortium identifies constructs which are suitable for capturing, specifying, modelling and testing real-time requirements. Based on these constructs, the consortium proposes time extensions to the formal languages as ITU-T recommendations. In the second project stage, tools will be developed which include the new time constructs.

Taking existing test generation tools as reference implementations, this paper evaluates the implications of existing and proposed time extensions to CATG.

* Part of this work has been sponsored by the European Commission under contract IST-1999-11557.

R. Reed and J. Reed (Eds.): SDL 2001, LNCS 2078, pp. 168–181, 2001.

It is structured as follows: Section 2 shows when and why time constructs are needed during testing. Section 3 contains an overview of the timer concepts in MSC, SDL and TTCN. In Sect. 4, timer support of the test generation tools TestComposer and Autolink is discussed. Section 5 is the main part of this paper. It examines first how the test generation process may be improved through the use of SDL-2000 together with the proposed extensions. Second, the benefits of using MSC-2000 for test purpose description are shown and a concrete mapping of MSC-2000 time concepts to TTCN-3 is presented. Section 6 concludes this paper.

2 Timer in Test Purpose Descriptions

Timers in test sequences have one of the following purposes:

- assuring that test cases end even if they are blocked due to unexpected behavior of the system under test (SUT);
- checking constraints on the response time of the SUT;
- delaying the sending of messages to the SUT in order to
  - allow the SUT to get into a state where it can receive the next signal (if the tester is too fast);
  - check the reaction of the SUT if a signal is delayed too long (invalid behavior specification);
  - check that the SUT does not send any signal for a given amount of time.

To guarantee the conclusion of a test case, one or more *global timers* are used. In the case of a single-tester test architecture, one timer is started at the beginning of test case execution. Its duration is chosen to be longer than the expected execution time of the test case. At the end of each possible test sequence, the timer is reset. In the exception handling section of the test case, the timeout of the global timer is caught and handled. If a distributed test system is used, a global timer is started within each test component participating in the test execution. In the case of a timeout in any test component, the other test components have to be notified in order to let them conclude the test execution gracefully.

Time constraints are checked through the use of one or a pair of *guarding timers*. Guarding timers are started when a signal is sent to the SUT. If a lower bound is specified in the time constraint, one timer has to expire before the response signal from the SUT is received. The second timer — which checks the upper bound of the time constraint — is reset immediately upon reception of the response signal. Premature reception of the response signal or the expiration of the second timer is caught in the exception handling section of the test case and result in a FAIL verdict.

A *delaying timer* is specified by inserting a timer start operation immediately followed by a timeout event into the test sequence.