*TIMED*TTCN-3 Based Graphical Real-Time Test Specification

Zhen Ru Dai, Jens Grabowski, and Helmut Neukirchen

Institute for Telematics, University of Lübeck
Ratzeburger Allee 160, D-23538 Lübeck, Germany
{dai,grabowsk,neukirchen}@itm.uni-luebeck.de
Tel.: +49 451 500 3721, Fax: +49 451 500 3722

Abstract. The textual *Testing and Test Control Notation* (TTCN-3) is frequently used in combination with *Message Sequence Chart* (MSC) and the MSC-based *Graphical Presentation Format for TTCN-3* (GFT). Both, MSC and GFT allow an automatic generation of TTCN-3 test case descriptions.
*TIMED*TTCN-3 is an extension of TTCN-3 for testing real-time properties and has been submitted for standardization. For a complete integration of *TIMED*TTCN-3 into the TTCN-3-based testing process, the usage of *TIMED*TTCN-3 in combination with MSC and GFT needs to be established.
This paper presents our approach for graphical real-time test specification based on MSC and *TIMED*GFT, which is our real-time extension of GFT. We explain how MSC can be used for the description of real-time test purposes and define *TIMED*GFT. Our approach includes the automatic generation of *TIMED*TTCN-3 test cases based on MSC test purposes and *TIMED*GFT diagrams.

1 Introduction

The *Testing and Test Control Notation* (TTCN-3) [8,13] of the *European Telecommunications Standards Institute* (ETSI) becomes more and more popular for test specification and test implementation. Several TTCN-3 tools supporting the design, documentation, compilation and execution of test campaigns are under development or are already commercially available [4,5,22,23]. Trials have shown that TTCN-3 is a modular and flexible test language that can be used effectively for testing numerous applications like, e.g., SIP [17], IPv6 [21], IDL-based interfaces [6] or XML-based interfaces [19]. Like the previous edition of TTCN-3, in the following called TTCN-2++ [7], TTCN-3 focusses on the description of test cases for functional black box testing.

The need for testing non-functional properties has been discussed in literature and several real-time and performance extensions for TTCN-2++ have been proposed [18,24]. Based on these experiences *TIMED*TTCN-3, [3] a real-time extension for TTCN-3, has been developed and submitted for standardization. *TIMED*TTCN-3 provides the concepts for testing hard real-time requirements.

D. Hogrefe and A. Wiles (Eds.): TestCom 2003, LNCS 2644, pp. 110–127, 2003.

The test of soft real-time properties, e.g., arrival or loss rates, may be realized in TTCN-3 by executing identical test cases several times or by using external load generators that are controlled by TTCN-3 test components.

In most cases, the testing process is not only based on one, but on several languages: Graphical languages like *Message Sequence Chart* (MSC), the *Specification and Description Language* (SDL) or the *Unified Modelling Language* (UML) are used as a basis for test generation, test specification or test visualisation [1,10,15,20] and data languages like the *Interface Definition Language* (IDL) or the *Abstract Syntax Notation One* (ASN.1) define the information to be exchanged between the *System Under Test* (SUT) and the test system during the test execution.

The usage of MSC [25] in functional black-box testing, including the standardized OSI conformance testing, as a graphical language for test purpose specification and test behaviour visualization has been thoroughly investigated by several authors [1,2,12,15]. MSC is very well suited for the specification of test purposes and allows an automated generation of TTCN-2++ test cases. TTCN-3 is backwards compatible with TTCN-2++ and thus, an automated generation of TTCN-3 test cases is also possible. The visualization of TTCN-2++ and TTCN-3 test cases by means of MSC is more problematic. The reason is that MSC does not provide the right level of abstraction. Test cases are implementations that are written for specific test configurations and include implementation details that cannot be represented easily by MSC diagrams. Furthermore, lots of test information is related to test data and MSC does not support a graphical representation of data. To overcome this problem, the MSC-based *Graphical Presentation Format for TTCN-3* (GFT) [9] focusses on the presentation of test behaviour and extends MSC with special constructs that emphasize test and TTCN-3 specific concepts in behaviour definition, e.g., setting of test verdicts or activation and deactivation of default behaviour. The GFT extensions of MSC are defined as shorthand notations for (in some cases more complex) MSC constructs, i.e., GFT is compatible to MSC.

This paper is about the usage of MSC and GFT for real-time testing. We discuss the specification of real-time test purposes by means of MSC and explain how *Timed*TTCN-3 test cases can be generated from MSC real-time test purposes. For the visualization of *Timed*TTCN-3 test cases we present a real-time extension for GFT (*Timed*GFT), which allows a graphical presentation of *Timed*TTCN-3 concepts. For simplicity, all concepts and extensions are explained by using the same simple Inres-based example, which is used in [3] to explain *Timed*TTCN-3.

2 Foundations

In this section, we describe the usage of MSC and GFT in the TTCN-3-based testing process, explain the principles of *Timed*TTCN-3 and summarize the time concepts of MSC. For the complete understanding of this paper, a basic knowledge of MSC, GFT and the TTCN-3 core notation is required.

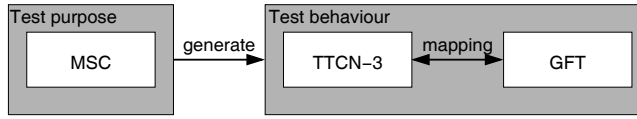


Fig. 1. MSC test purposes, TTCN-3 test behaviour and GFT diagrams

2.1 Test Purpose Definition and Test Behaviour Visualization

The difference between test purposes described in form of MSCs and a test behaviour[1] visualized by GFT diagrams are the different levels of abstraction and points of view. The relations between test purposes and the behaviour of test cases are shown in Fig. 1.

An MSC test purpose is an abstract description of a test. It describes the test from the perspective of the SUT and makes no assumptions about the implementation of the test configuration. By providing information about the test configuration, it is possible to generate TTCN-3 test behaviours. Test behaviour descriptions define tests from the perspective of the test system. They are written for a specific test configuration and include all the activities to coordinate the test components. A TTCN-3 test behaviour description can be visualized in form of a GFT diagram, i.e., there exists a bidirectional mapping between a TTCN-3 test behaviour and a corresponding GFT diagram.

An example may clarify these differences. We want to test an Initiator implementation of the Inres protocol [14]. The SUT can be accessed by using the interfaces ISAP and MSAP. The test purpose is the test of 100 data transfers. For doing this, a connection needs to be established and after the test, the connection has to be released. A formalization of this test purpose in form of an MSC is shown in Fig. 2b[2].

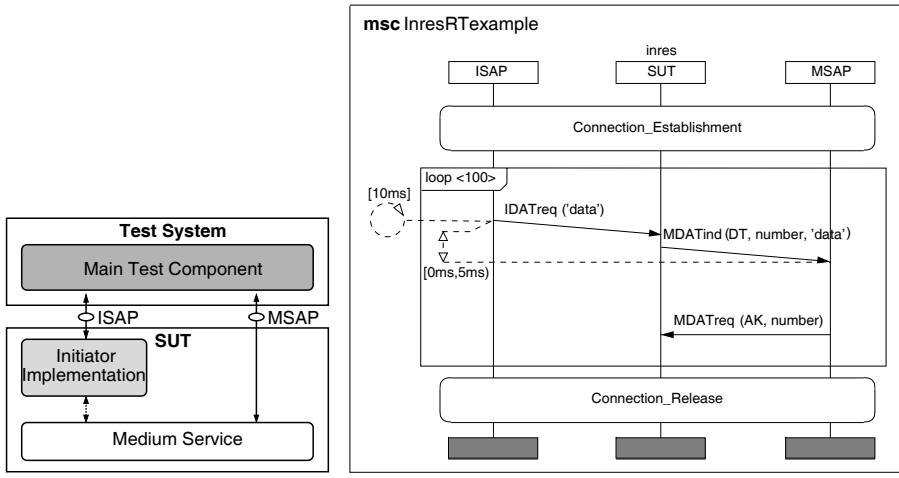
The MSC describes the test purpose from the point of view of the SUT, i.e., only the required information exchange at the ISAP and MSAP is shown. It includes no assumptions about implementation of the test system, e.g., the number of test components and the required synchronization among test components are not specified.

The test case InresRTexample shown in the lines 6-29 of Fig. 3[3] is generated automatically from our test purpose example (Fig. 2b) and the test configuration presented in Fig. 2a. The test configuration only consists of one *Main Test Component* (MTC), which controls both interfaces of the SUT. The test case InresRTexample can be visualized by the GFT diagram shown in Fig. 8[2]. A simple comparison of GFT diagram and textual TTCN-3 description shows that there exists a bi-directional mapping between text and graphics.

[1] In the following, the term *test behaviour* refers to TTCN-3 test cases, altsteps, functions and module control.

[2] The meaning and usage of the MSC time constructs, i.e., the dashed lines and arrows with annotations, in Fig. 2b and Fig. 4 will be explained in the following sections.

[3] The *TIMED*TTCN-3 statements, e.g., **now** and **resume**, in Fig. 3 are explained in the following sections.



(a) Test configuration (b) Real-time test purpose

Fig. 2. Test architecture and test purpose for the Inres test case example

2.2 *Timed*TTCN-3

*Timed*TTCN-3 has been defined in [3]. It introduces the concept of absolute time into TTCN-3, provides a means to specify time synchronized test components, extends the TTCN-3 logging mechanism, supports online and offline evaluation of tests and adds the new test verdict **conf** to the existing TTCN-3 test verdicts.

Absolute time is introduced by means of clocks. Each test component has an associated local clock, which can be read by using a **now** operation (lines 13, 19 and 22 in Fig. 3). In addition, a **resume** statement can be used to wait until a certain point in time (line 16). For example, the statement **resume**(**self.now**+3.0) defines that the following statement or operation will be performed 3 seconds after the invocation of the **resume** statement. Time is represented as a **float** value that represents the number of seconds counted from a fixed starting point. The starting point is considered to be test system specific and therefore, not defined by *Timed*TTCN-3.

*Timed*TTCN-3 allows to specify time synchronized test components by means of *timezones*. A timezone is an (optional) attribute, which can be assigned to a test component when the component is created. Test components with the same attribute are considered to be synchronized, i.e., they have the same absolute time. Components without timezone attribute are considered to be not time synchronized with other components.

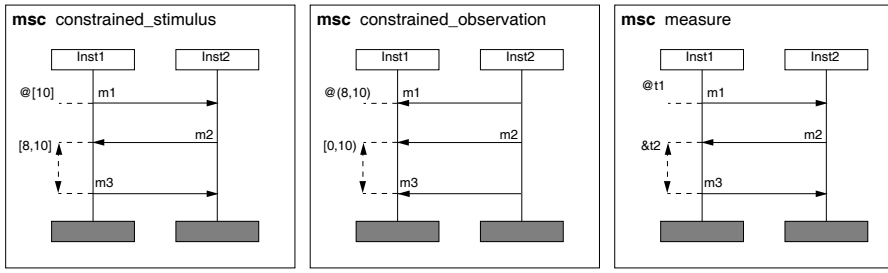
The TTCN-3 logging mechanism is improved by allowing to put TTCN-3 data structures into the the logfile and by giving access to the logfile in module control after test case termination. This allows to store timestamps of time critical test events as records in the logfile by using **log** statements (lines 19 and 22 in Fig. 3).

*Timed*TTCN-3 distinguishes between online and offline evaluation of real-time properties. Online evaluation refers to an evaluation during test execution,

```
(1) module InresRTexample_module() {
(2)   type record TimestampType {
(3)   float logtime,
(4)   charstring id
(5)   }
(6) testcase InresRTexample() runs on inres {
(7)   var float sendTime1:=-1.0;
(8)   var integer iterator1:=0;
(9)   var default OtherwiseFailDefault:=activate(OtherwiseFailAltStep);
(10)  Connection_Establishment();
(11)  for (iterator1:=0; iterator1<100; iterator1:=iterator1+1) {
(12)    if (sendTime1==-1.0) {
(13)      sendTime1:=self.now+0.01;
(14)    }
(15)    else {
(16)      resume(sendTime1);
(17)      sendTime1:=sendTime1+0.01;
(18)    }
(19)    log(TimestampType:{self.now,"IDATreq1"});
(20)    ISAP.send(IDATreq:{"data"});
(21)    MSAP.receive(MDATind:{DT,number,"data"});
(22)    log(TimestampType:{self.now,"MDATind2"});
(23)    MSAP.send(MDATreq:{AK,number});
(24)  }
(25)  Connection_Release();
(26)  deactivate(OtherwiseFailDefault);
(27)  setverdict(pass);
(28)  stop;
(29) }
(30) control {
(31)  var testrun myTestrun;
(32)  var logfile myLog;
(33)  var verdicttype myVerdict;
(34)  myTestrun:=execute(InresRTexample);
(35)  myVerdict:=myTestrun.getverdict;
(36)  if (myVerdict==pass) {
(37)    myLog:=myTestrun.getlog;
(38)    myVerdict:=evalMultipleDelays("IDATreq1","MDATind2",
                                      0.0,incl,0.005,excl,myLog);
(39)    myTestrun.setverdict(myVerdict);
(40)    }
(41)  }
(42) }
```

Fig. 3. *Timed*TTCN-3 Inres test case generated from the test purpose in Fig. 2

i.e., the final test verdict is calculated during the test, whereas offline evaluation means that the final evaluation is done after test case termination. Offline evaluation is realized by giving access to the logfile produced by a test case execution.



(a) Time constraint (b) Time constraints (c) Time measurements

Fig. 4. MSC time constructs

For this, a testrun handle (line 31 in Fig. 3) has been introduced, which gives access to the logfile (line 32) and the final verdict of the functional test execution (line 35). Special functions allow to sort, read and navigate in a logfile. A logfile may be passed into a library function (line 38) that performs the evaluation of the logfile and calculates the final verdict of the TIMEDTTCN-3 real-time test case. The final verdict can be assigned by using **setverdict** operation (line 39).

The additional test verdict **conf** has been introduced by TIMEDTTCN-3 as an indication that a test case executed correctly with respect to the functional behaviour but failed with respect to a non-functional property. The existing verdict **pass** is used to indicate success with respect to both criteria.

2.3 Time Constructs in MSC

MSC allows to attach time annotations to events. MSC distinguishes between *time constraints* and *time measurements*.

Time constraints are shown in Fig. 4a and 4b. They can be either absolute, i.e., refer to the absolute time of occurrence of one single event, or relative, i.e., constrain the duration between two events. The values of the constraint is specified using intervals. The interval boundaries may be open or closed. An open boundary is indicated by a parenthesis, i.e., '(' or ')', and a closed boundary is defined by a square bracket, i.e., '[' or ']'. An omitted lower bound is treated as zero, an omitted upper bound as ∞. A single value in square brackets represents an interval which contains just that single element. If a time annotation defines no time unit, the default unit *seconds* is used.

Time measurements are presented in Fig. 4c. A measurement may measure the absolute time when a single event occurs or the relative delay between two events. The value of a measurement is stored in a variable which is local to the instance that owns that variable.

In addition to a relative time constraint, Fig. 2 contains a time constraint for a cyclic event (sending message IDATReq("data") every 10ms) inside a loop inline expression. The definition of such periodic events is not supported in the MSC standard. Therefore, we use an extension proposed in [16]. An alternative extension with similar expressive power has been presented in [26].

3 Generating *Timed*TTCN-3 from MSC Test Purposes

Using MSC for test purpose specification and the consecutive test case generation as suggested in this paper is not new and has already been implemented in several academical and industrial tools like, e.g., *Autolink*, *Testcomposer* or *ptk*.

Autolink [20] and *Testcomposer* [15] support the generation of TTCN-2++ test cases either from SDL specifications and MSC test purposes or directly from MSC test purposes. Both tools focus on the generation of tests for testing functional requirements.

The *ptk* tool [1] also generates TTCN-2++ test cases from MSC test purposes. The test generation for real-time constraints, which can be implemented by the TTCN-2++ timer mechanism, is supported by non-standard time annotations in the MSC test purposes. *Autolink* and *ptk* also support the generation of test cases for concurrent test architectures [1,11].

None of the tools mentioned above is currently able to generate TTCN-3 output. Therefore, we developed an MSC to TTCN-3 generator as an internal case study. While past versions only accepted MSC test purposes containing functional requirements, the latest prototype supports also the real-time constructs described in Section 2.3. Our current prototype does not support concurrent test architectures, i.e., the prototype generates test cases with one main test component (MTC) only. For future versions, we plan to support concurrent test architectures by implementing the approach described in [11].

In the following, we explain how MSC can be used for specifying real-time test purposes and how these test purposes can be automatically transformed into *Timed*TTCN-3 test cases. For simplicity, we restrict ourselves to non concurrent test architectures. For concurrent test architectures, the synchronization among test components has also to be taken into consideration.

3.1 Specification of MSC Real-Time Test Purposes

We use the common practice to specify test purposes using *system level* MSCs. A system level MSC has one designated instance representing the SUT. All other instances correspond to the different interfaces of the SUT. Together with the contained message exchange from and to the interfaces of the SUT, this information can be used to derive corresponding test cases.

In contrast to other approaches (e.g., [1]), we do not generate one test case for each of the traces which is possible due to the interleaving semantics of MSC. Instead, we extract just one single representative *path* from the test purpose MSC, which takes the queue semantics of TTCN-3 into consideration. This will be explained in the next section.

For the specification of MSC real-time test purposes, the MSC real-time constructs described in Section 2.3 are used. We allow to attach MSC time constraints and measurements (see Fig. 2b) to the communication events along the instances that represent the interfaces of the SUT.

While the meaning of allowed MSC time measurements (Fig. 4c) in test purpose descriptions is straightforward (i.e., observation of the point in time

when an event occurs and its storage in a *Timed*TTCN-3 variable), MSC time constraints can be used with two different aims: They may either describe a timely stimulation of the SUT (*time constrained stimulus*) or a response from the SUT shall arrive within a certain period of time (*time constrained observation*). Both cases look similar, but can be distinguished as follows:

The test system should perform a time constrained stimulus if an absolute time constraint is attached to a send event, or a relative time constraint is attached to a pair of events, where the second event is a send event[4] (cf. Fig. 4a, where Inst2 represents the SUT).

The test system should perform a time constrained observation, if an absolute time constraint is attached to a receive event, or a relative time constraint is attached to a pair of events, where the second event is a receive event[4] (cf. Fig. 4b, where Inst2 represents the SUT).

In our test purpose example (Fig. 2b), both types of relative time constraints are used. The cyclic real-time requirement constraint can be unrolled to a sequence of relative time constraints and requires to send a stimulus every 10ms. The other constraint ([0ms,5ms)), which is attached to the messages IDATreq and MDATind, describes a passive observation of MDATind.

An MSC time measurement and an MSC time constraint used for time constrained observation, result in a similar test behaviour. In both cases the current time is acquired. But, only observing time constraints allow to attach the actual intervals of real-time requirements to the graphical symbols. This identifies them as real-time requirements. In contrast, a real-time requirement cannot be specified by MSC time measurements, because they only allow to specify names for observations, but no concrete time values. As a result of these considerations, MSC time constraints should be the preferred means for specifying real-time test purposes. Nevertheless, MSC time measurements may be useful for gathering time information, which is re-used later as part of other MSC time constraints.

3.2 Transformation of Functional Concepts from MSC to TTCN-3

The transformation of the static aspects of an MSC test purpose into TTCN-3 is very simple. The interfaces of the SUT are described by MSC instances. They are mapped to TTCN-3 ports. For the mapping of MSC events to TTCN-3 statements, only the events along the interface instances are relevant. MSC send events are mapped to TTCN-3 send operations and MSC receive events are mapped onto TTCN-3 receive operations. The MSC message names refer to TTCN-3 data types or signature definitions. MSC message parameters, refer to TTCN-3 templates or define inline templates.

Configuration and data descriptions, like the definition of port types, component types, data types, signatures and templates, cannot be generated automatically from MSC test purposes. They have to be specified manually or imported from other TTCN-3 modules.

[4] The type of the first event involved in the relative time constraint is irrelevant, since the time constraint is essentially imposed on the second event.

An MSC test purpose specification may also include references and the inline expressions **alt**, **loop** and **par**. We consider each usage of a reference or an inline expression as one single event, which in case of references or inline expressions may include partially ordered events. This means, these constructs are synchronization points, even though this violates the official MSC semantics [25]. The MSC semantics assumes a weak sequential composition semantics. However, many test case specifiers regard this as counter-intuitive. Hence, the corresponding TTCN-3 constructs suggested in our mapping do not allow that sort of interleaving. Our transformation algorithm maps MSC references to TTCN-3 function calls, MSC **alt** inline expressions to TTCN-3 **alt** statements, MSC **loop** inline expressions to TTCN-3 **for** statements and MSC **par** inline expressions to TTCN-3 **interleave** statements.

Our generation algorithm uses the *Autolink* approach [20] for the calculation of the control flow of TTCN-3 test cases[5]. The algorithm computes a so-called *path* from the partially ordered set of MSC events. Such a path specifies a set of traces, which includes no nondeterminism due to non-receiving events, but considers all interleavings due to receiving events. This path representation takes the port queue semantics of TTCN-3 into consideration. A path can be visualized in form of a tree, where branching is related to alternative receiving events. Our TTCN-3 generation algorithm computes a TTCN-3 test case that allows to test all sequences of events described by the corresponding path.

The *TIMED*TTCN-3 test case shown in the lines 6-29 of Fig. 3 has been generated automatically from the MSC test purpose shown in Fig. 2b. A comparison of both figures may indicate how functional concepts in MSCs are transformed into TTCN-3.

In this paper, we only have space to present the basic ideas of our TTCN-3 generation procedure. For test purpose specification, we also support HMSCs and allow the usage of timers, actions and conditions in MSC test purposes. The treatment of these constructs and the handling of complex MSC test purposes is investigated in [1,11,20]. Our implementation follows these approaches.

3.3 Transforming MSC Real-Time Constructs to *TIMED*TTCN-3

For deriving test cases, the different usages of real-time constructs in MSC test purposes identified in Section 3.1 have to be handled separately.

Time Constrained Observations.
Time constrained observations in test purposes are translated to *TIMED*TTCN-3 by generating timestamps for the observed events. For offline evaluation, the timestamps are stored in the logfile produced by the test case. In the online evaluation approach timestamps are compared during the test run, because they are stored in variables. In the following, we focus on offline evaluation, but all

[5] Both, TTCN-2++ and TTCN-3 support asynchronous communication over message queues. Therefore, the *Autolink* approach for the generation of TTCN-2++ test cases from MSC test purposes can also be used for the generation of TTCN-3 test cases.

considerations can be generalized to online evaluation, because both approaches are based on the generation of timestamps.

The example test purpose in Fig. 2b contains a time constrained observation. Since the constraint is attached to the messages IDATreq and MDATind, the *Timed*TTCN-3 statements for generating timestamps (lines 19 and 22 of Fig. 3) are placed directly before and after the associated communication operations (lines 20–21).

Our TTCN-3 generator generates timestamps that contain the value of the local clock when the timestamp is generated and a label of type **charstring**, which is used to identify timestamps afterwards. The value of the label is generated from the message name used in the MSC plus a consecutive number to distinguish between different occurrences of the same message. The type definition for the TimestampType is added automatically at the top of the generated module (lines 2–5). At the bottom, i.e., in the control part, a call to an evaluation function is added (line 38) for each time constrained observation defined in the MSC test purpose.

Evaluation functions may be provided in form of *Timed*TTCN-3 libraries. For example, the predefined evaluation function evalMultipleDelays is used to retrieve and evaluate a sequence of matching pairs of timestamps from a given logfile. The parameters of the function identify the timestamps to be compared, define the time interval between two timestamps[6] and refer to the logfile, which should be analyzed. If the time difference of each timestamp pair found in the logfile fulfills the given requirement, the verdict **pass** is returned. If the real-time requirement is violated, **conf** is returned and indicates a non-functional failure. This result contributes to the final verdict of the test case (line 39).

Placement of Timestamping Statements.
In a perfect world, no time passes between a send or receive event and the corresponding timestamp generation. Hence, the time stored in the timestamp is the actual time of the event. However, test cases are implemented on real hardware, some time passes between both statements. Thus, we have the choice of putting the **log** statement before or after a time constrained event derived from a test purpose. For **receive** operations, we have to put the **log** statement after the **receive** (lines 21–22 in Fig. 3), because a **receive** operation is blocking. But for a **send** operation, we have the option to put a timestamping **log** statement before or after the **send** operation.

In our example, for the time constraint related to the messages IDATreq and MDATind, the **log** statement associated to the **send** operation in line 20 of Fig. 3 may be inserted before or after this **send** operation. In the first case, the observed duration would be longer than in the second case. The choice of placement depends on whether the time constraint stipulates a minimal or maximal duration.

[6] Upper and lower bound of the interval are defined by two **float** values. The parameters incl and excl define whether a boundary is closed or open.

```
(1) float sendTime;
(2) port.receive / port.send
(3) sendTime:=self.now+d;
    ...
(4) resume(sendTime);
(5) port.send
```

Fig. 5. *Timed*TTCN-3 skeleton for timed stimulus

If the time constraint only has an upper bound (or the lower bound is zero as in e.g., [0ms,5ms)), a maximal duration is specified. In this case, choosing the placement which yields the shorter observed duration might result in a **pass** verdict even though the actual duration violated slightly the real-time constraint. Instead, we place in this case the **log** statement before the **send** operation as shown in lines 19–20.

The opposite considerations hold for testing minimal durations, e.g., (1ms,). We are on the safe side with an observed duration which is shorter than the actual duration. Thus, we insert the **log** statement after the **send** operation.

In the combined case[7], i.e. neither the lower interval bound is omitted or zero nor the upper bound is omitted (e.g. [8ms,10ms]), it is a matter of preference which bound is given the priority. If we assume that encoding for sending and decoding for receiving takes a similar amount of time, we put the **log** statement after the **send** operation because we have to put it also after the **receive** operation. This may lead to a measurement, which is closer to the reality since the extra delay introduced by both operations will eliminate each other.

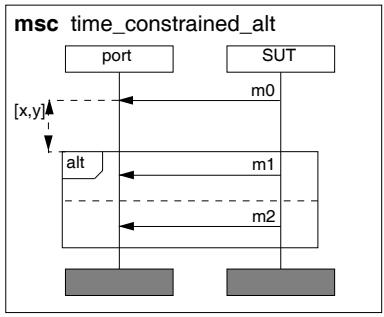
Time Constrained Stimuli.
A time constrained stimulus in an MSC test purpose description is translated into a *Timed*TTCN-3 **resume** statement, which is used to schedule the execution of the related **send** operation. A generic *Timed*TTCN-3 skeleton for a time constrained stimulus is shown in Fig. 5.

If the time constraint consists of a single point in time, e.g., [d], this value can be used as relative offset to the **self.now** expression in line 3 of Fig. 5.

If the time constraint is an interval, e.g., [x,y], any of the values inside the interval is possible as delay of the **send** operation. This may lead to an infinite number of test cases, which is infeasible in practice. Using test data selection heuristics, like domain boundary analysis, an appropriate number of test cases can be selected, e.g., d=x for testing the extreme lower and d=y for testing the extreme upper allowed point in time.

Time Constraints Attached to Inline Expressions and References.
For MSC inline expressions and references several special cases have to be considered. In this paper, we are only able to cover a few of them.

[7] Note, that specifying a single element as interval for a time constrained observation is not recommended, because it is very unlikely, that exactly that value is matched.



(a) Test purpose

```
(1) port.receive(m0);
(2) log(TimestampType:{self.now,"m0"});
    ...
(3) alt {
(4)   [ ] port.receive(m1); {
(5)       log(TimestampType:
              {self.now,"Either_m1_or_m2"}); }
(4)   [ ] port.receive(m2); {
(5)       log(TimestampType:
              {self.now,"Either_m1_or_m2"}); }
(6) }
```

(b) Derived test case

Fig. 6. Time constrained observation attached to first event of alternative

A cyclic time constraint contained in a loop, can be treated nearly like ordinary relative time constraints: In a time constrained cyclic observation, timestamps are not generated for a pair of two communication events, but for a sequence of a single communication event. Hence, a call to a different evaluation function, working on sequences of a single timestamp only, is necessary. For a time constrained cyclic stimulus, a different set of statements than presented in Fig. 5 is required: the first execution of the **send** operation has to be performed immediately, while all subsequent executions need to adhere to the cyclic time constraint. This is achieved by the statements given in lines 7, 12–18 of Fig. 3.

For MSC inline expressions and references, MSC allows to impose time constraints on the first or last occurring MSC event which is contained in the inline expression or reference, respectively. This means, that for every possible first or last event, a timestamp has to be generated. Since a magnitude of different permutations is possible, we present just the example in Fig. 6.

4 *Timed*GFT

GFT [9] is one of the standardized presentation formats of TTCN-3. It provides an exact way of displaying TTCN-3 behaviour descriptions, i.e., test cases, alt-steps, functions and module control, graphically. An one-to-one mapping from TTCN-3 behaviour descriptions to GFT diagrams and vice versa can be found in the appendices C and D of [9].

GFT is based on the MSC standard. It uses a subset of MSC and extends this subset with test specific symbols and keywords. For all TTCN-3 statements, there exists an appropriate GFT symbol. Thus, only the real-time extensions of *Timed*TTCN-3 have to be considered in order to define our real-time extension of GFT, which is called *Timed*GFT. The *Timed*GFT presentation of the special *Timed*TTCN-3 concepts and statements is summarized in Fig. 7.

4.1 Timezones

For the specification of clock synchronized test components, *Timed*TTCN-3 introduced the concept of timezones. A timezone is an attribute, which is assigned

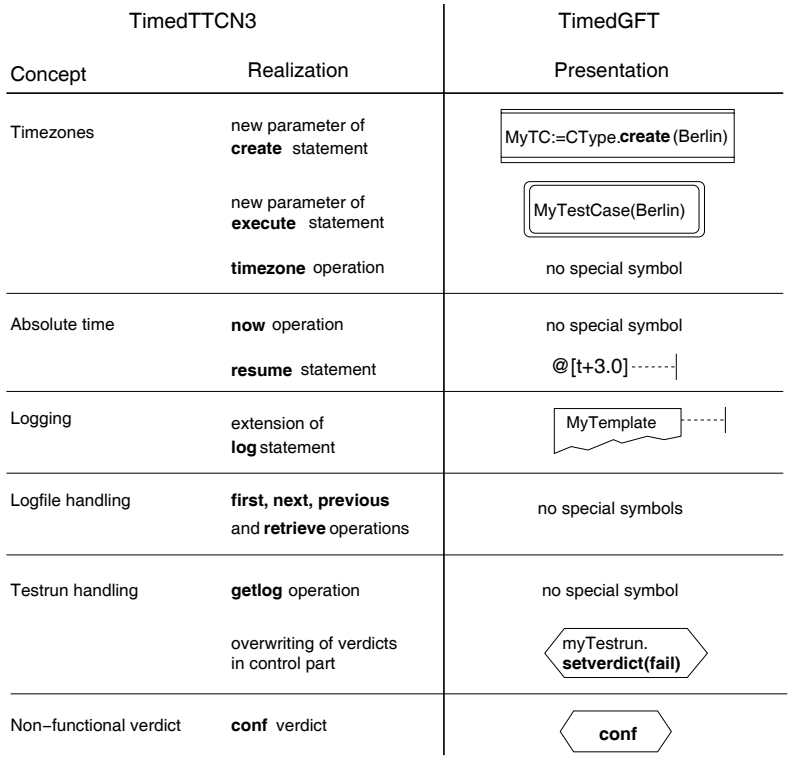


Fig. 7. Real-time constructs of *TIMED*GFT

to a test component during its creation. In *TIMED*GFT, the creation of a test component is related to create and execute symbols. The assignment of a timezone is an additional parameter in a create or execute symbol (Fig. 7).

A test component may read its timezone attribute by means of a **timezone** operation. *TIMED*GFT provides no special symbol for the **timezone** operation. Depending on the usage, the **timezone** operation may appear in several symbols. For example, a **timezone** operation will appear in an action box, if it is used in an assignment, or a **timezone** operation will be presented within a reference symbol, if it defines the actual parameter of an altstep or function call.

4.2 Absolute Time

*TIMED*TTCN-3 supports the usage of absolute time by providing the **now** operation and the **resume** statement.

The *now* Operation.
The **now** operation is used for the retrieval of the current local time of a test

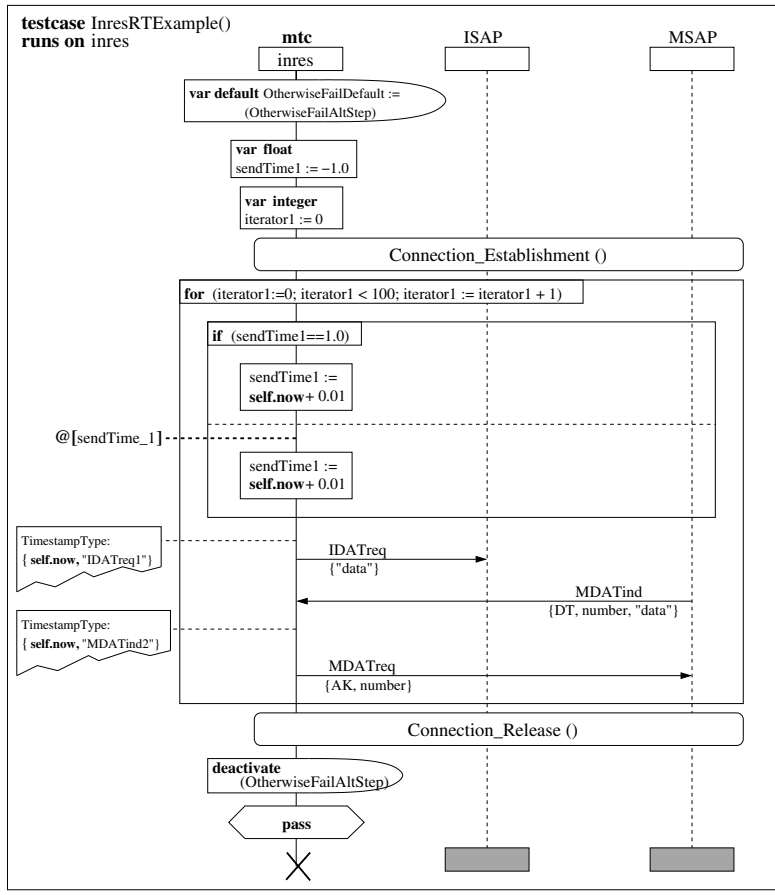


Fig. 8. TIMEDGFT presentation of the Inres test case in Fig. 3

component. The local character of the **now** operation is reflected by its application to the **self** handle. TIMEDGFT provides no special symbol for the **now** operation. Depending on its usage, the **now** operation appears in different symbols. In Fig. 8, the **now** operation is used in assignments and in **log** statements. Therefore, they appear as inscriptions in action boxes and log symbols.

The *resume* Statement.
The **resume** statement provides the ability to delay the execution of a test component until a specified absolute time is reached. TIMEDGFT has adopted the absolute time constraint symbol of MSC to present the **resume** statement graphically. In Fig. 8, the visualization of **resume**(sendTime1) can be found. Contrary to MSC, the dashed time line is attached to an instance and not to an event. The reason is that the **resume** statement is a statement of its own and can not be related to other events. For the mapping of TIMEDGFT to MSC, the

dashed time line may be attached to the event, which is deferred by the **resume** statement[8].

4.3 Logging Mechanism

*Timed*TTCN-3 refines the TTCN-3 **log** statement by allowing to write structured *Timed*TTCN-3 timestamp data values into logfiles. In GFT, **log** statements are presented in action boxes. *Timed*GFT introduces a new log symbol. The reason for this new symbol is that we want to emphasize places in the test behaviour, where time and other information are collected in the logfile. Figure 8 presents the new symbol. It is used for logging the two timestamps {**self**.**now**, "IDATreq1"} and {**self**.**now**, "MDATind2"} of type TimestampType.

4.4 Testrun and Logfile Handling

The result of the execution of a *Timed*TTCN-3 test case is a test verdict and a log file. In the module control part, *Timed*TTCN-3 gives access to both by a testrun handle, which is returned by the **execute** statement.

For the graphical presentation of a module control part, GFT provides a control diagram, which includes one control instance only. Fig. 9 shows the control diagram of the example Inres *Timed*TTCN-3 module.

The **getlog** operation can be applied to a testrun handle in order to access the log file, e.g. for offline evaluation. The context of the **getlog** operation determines the symbol in which it is presented. For example, in Fig. 9, the **getlog** operation is used in an assignment and therefore, presented in an action box.

The actual evaluation function evalMultipleDelays, which might be imported from a library, is not shown. Such functions typically operate on logfiles. In order to sort and navigate in logfiles, *Timed*TTCN-3 provides the functions **first**, **next**, **previous** and **retrieve**. They have no special *Timed*GFT presentation. In the same manner as for the **getlog** operation, their presentation depends on the context in which they are used.

4.5 Verdict Handling

In addition to the existing verdicts, *Timed*TTCN-3 extends the verdict handling of TTCN-3 by introducing the additional verdict **conf** and by allowing to access and overwrite a test verdict in the control part of a module.

The handling of verdicts in *Timed*GFT is almost identical to their handling in GFT. The **setverdict** operation is always presented in a condition symbol (bottom of Fig. 9). There exists no special symbol to emphasize the **getverdict** operation. The context determines the symbol in which it is presented.

The only difference of *Timed*GFT to GFT with respect to verdict handling is the setting of verdicts in the module control part or in functions called by

[8] The mapping of GFT to MSC is defined in Annex F of [9]. A similar mapping of the *Timed*GFT to MSC will be defined when *Timed*GFT becomes part of GFT.

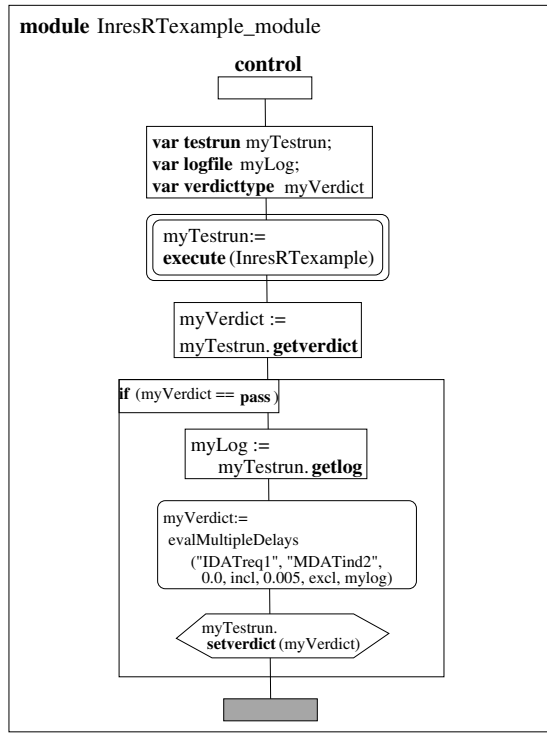


Fig. 9. TIMEDGFT control diagram for Fig. 3

the module control part. In these cases, the complete TIMEDTTCN-3 statement including the testrun handle has to be written into the condition symbol, because without the testrun handle, the relation between verdict and testrun would not be clear. The retrieval and the setting of a verdict by means of a testrun handle inside a control diagram is shown in Fig. 9.

4.6 Evaluation Functions

On- and offline evaluation involves the application of different pre-defined functions and operators which operate on the gathered timestamps. TIMEDGFT provides no special symbols for the call of evaluation functions. They may be presented in action or reference symbols.

5 Outlook

In this paper, we presented the MSC-based specification of real-time test purposes and the generation of TIMEDTTCN-3 test cases from MSC test purposes. Furthermore, we introduced TIMEDGFT, a real-time extension of GFT, which

allows to present *Timed*TTCN-3 graphically. This paper can be seen as the continuation of our work on *Timed*TTCN-3 [3].

At present, we develop tools, which support the generation of *Timed*TTCN-3 test cases from MSC test purposes. A first prototype of our tool is already available and our experiments with more complex test purposes are promising. Furthermore, we work on the prototype of a *Timed*GFT tool.

We also started to define the semantics of *Timed*TTCN-3 by enriching the semantics of TTCN-3 with real-time concepts. We plan to submit this semantics extension together with *Timed*GFT to the ETSI standardization process in order to have a full integration of *Timed*TTCN-3 into the TTCN-3 standards series.

References

1. P. Baker, P. Bristow, C. Jervis, D. King, and B. Mitchell. Automatic Generation of Conformance Tests From Message Sequence Charts. In *Proceedings of the 3rd SAM (SDL and MSC) Workshop*, 2002.
2. P. Baker, E. Rudolph, and I. Schieferdecker. Graphical Test Specification – The Graphical Format of TTCN-3. In R. Reed and J. Reed, editors, *SDL2001 – Meeting UML*. Springer, 2001.
3. Z.R. Dai, J. Grabowski, and H. Neukirchen. Timed TTCN-3 – A Real-Time Extension for TTCN-3. In I. Schieferdecker, H. König, and A. Wolisz, editors, *Testing of Communicating Systems*, volume 14, Berlin, March 2002. Kluwer.
4. Danet TTCN Toolbox – TTCN-3. `http://www.bss.danet.de/solution/ttcn/ttcn_toolbox_ttcn-3_uk.htm`, 2002.
5. Da Vinci Communications Terzo tools product information. `http://www.davinci-communications.com/products_ttcn3.html`, 2002.
6. M. Ebner, A. Yin, and M. Li. Definition and Utilisation of OMG IDL to TTCN-3 Mappings. In I. Schieferdecker, H. König, and A. Wolisz, editors, *Testing of Communicating Systems - Application to Internet Technologies and Services*, volume 14. Kluwer, 2002.
7. ETSI Technical Report (TR) 101 666 (1999-05): Information technology - Open Systems Interconnection Conformance testing methodology and framework; The Tree and Tabular Combined Notation (TTCN) (Ed. 2++). European Telecommunications Standards Institute (ETSI), Sophia-Antipolis (France), 1999.
8. ETSI European Standard (ES) 201 873-1 V2.2.1 (2002-08): The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. European Telecommunications Standards Institute (ETSI), Sophia-Antipolis (France), 2002.
9. ETSI European Standard (ES) 201 873-3 V2.2.1 (2002-09): The Testing and Test Control Notation version 3; Part 3: Graphical Presentation Format for TTCN-3 (GFT). European Telecommunications Standards Institute (ETSI), Sophia-Antipolis (France), 2002.
10. N. Goga. Comparing TorX, Autolink, TGV and UIO Test Algorithms. In R. Reed and J. Reed, editors, *SDL2001 – Meeting UML*. Springer, 2001.
11. J. Grabowski, B. Koch, M. Schmitt, and D. Hogrefe. SDL and MSC Based Test Generation for Distributed Test Architectures. In R. Dssouli, G. von Bochmann, and Y. Lahav, editors, *SDL'99 - The next Millenium*. Elsevier Science Publishers B.V., 1999.

12. J. Grabowski and T. Walter. Visualisation of TTCN test cases by MSCs. In Y. Lahav, A. Wolisz, J. Fischer, and E. Holz, editors, *Proceedings of the 1st Workshop of the SDL Forum Society on SDL and MSC - SAM'98*, 1998.
13. J. Grabowski, A. Wiles, C. Willcock, and D. Hogrefe. On the Design of the New Testing Language TTCN-3. In H. Ural, R.L. Probert, and G. von Bochmann, editors, *Testing of Communicating Systems*, volume 13. Kluwer, 2000.
14. D. Hogrefe. Report on the Validation of the Inres System. Technical Report IAM-95-007, Universität Bern, November 1995.
15. A. Kerbrat, T. Jéron, and R. Groz. Automated test generation from SDL specifications. In R. Dssouli, G. von Bochmann, and Y. Lahav, editors, *SDL'99 - The next Millenium*. Elsevier Science Publishers B.V., 1999.
16. H. Neukirchen. Corrections and extensions to Z.120, November 2000. Delayed Contribution No. 9 to ITU-T Study Group 10, Question 9.
17. I. Schieferdecker, S. Pietsch, and T. Vassiliou-Gioles. Systematic Testing of Internet Protocols - First Experiences in Using TTCN-3 for SIP. In *Proceedings of the 5th IFIP Africom Conference on Communication Systems, Cape Town (South Africa)*, May 2001.
18. I. Schieferdecker, B. Stepien, and A. Rennoch. PerfTTCN, a TTCN Language Extension for Performace Testing. In M. Kim, S. Kang, and K. Hong, editors, *Testing of Communicating Systems*, volume 10. Chapman & Hall, 1997.
19. I. Schieferdecker and B. Stepien. Automated Testing of XML/SOAP based Web Services. In *Proceedings of the 13th. Fachkonferenz der Gesellschaft für Informatik (GI) Fachgruppe "Kommunikation in verteilten Systemen" (KiVS), Leipzig (Germany)*, Feb. 26.–28. 2003.
20. M. Schmitt, A. Ek, J. Grabowski, D. Hogrefe, and B. Koch. Autolink – Putting SDL-based test generation into practice. In A. Petrenko and N. Yevtuschenko, editors, *Testing of Communicating Systems*, volume 11. Kluwer, 1998.
21. J.Z. Szabó. Experiences of TTCN-3 Test Executor Development. In I. Schieferdecker, H. König, and A. Wolisz, editors, *Testing of Communicating Systems - Application to Internet Technologies and Services*, volume 14. Kluwer, 2002.
22. Telelogic Tau/Tester product information. `http://www.tautester.com/`, 2002.
23. Testing Technologies TT Tool Series product information. `http://www.testingtech.de/products/TTToolSeries.html`, 2002.
24. T. Walter and J. Grabowski. A Framework for the Specification of Test Cases for Real Time Distributed Systems. *Information and Software Technology*, 41:781–798, 1999.
25. Recommendation Z.120: Message Sequence Charts (MSC). International Telecommunication Union (ITU-T), Geneve, 1999.
26. T. Zheng and F. Khendek. An extension to MSC-2000 and its application. In *Proceedings of the 3rd SAM (SDL and MSC) Workshop*, 2002.