Test Case Specification with Real-Time TTCN

Thomas Walter[a] and Jens Grabowski[b]

[a]Swiss Federal Institute of Technology Zurich, Computer Engineering and Networks Laboratory (TIK), CH-8092 Zurich, Switzerland, e-mail: walter@tik.ee.ethz.ch, http://www.tik.ee.ethz.ch/

[b]University of Lübeck, Institute for Telematics, Ratzeburger Allee 160, D-23538 Lübeck, Germany, e-mail: jens@itm.mu-luebeck.de, http://www.itm.mu-luebeck.de/

Abstract

The ever increasing dissemination of distributed real-time and multimedia applications makes testing of real-time constraints an important issue. The current conformance testing methodology does not define the means to cope with this new requirement. Especially, the test notation TTCN (Tree and Tabular Combined Notation) cannot be used to express and thus test real-time behaviour. In order to adapt the well established conformance testing methodology to this new need, we extend TTCN with real-time features. This is achieved by annotating TTCN statements with time intervals specifying the earliest and latest execution times. In this paper we introduce real-time TTCN and discuss its features and characteristics. To motivate our work, in the main part of the paper we evaluate real-time TTCN against TTCN and assess its suitability for testing real-time constraints by defining test cases for real applications. It turns out that without additional assumptions concerning the execution speed of test systems, TTCN cannot be used for testing real-time constraints.

1 Introduction

For years testing is known as a process for validating software, i.e., checking that software meets the expectations of its users [19]. The advantages and disadvantages of testing compared to rigorous formal verification are: Firstly, it is a generally applicable process. Secondly, it is a theoretically well understood process for which methods and techniques exist to derive test data from specifications. Thirdly, testing is a cost-effective process, if one considers the trade-off between resources spent and software quality achieved. One of the disadvantages is that testing does not prove absence of errors in the tested software.

Testing, or to be precise *conformance testing*, is also the generally applied process in validating communication software. A conformance testing methodology and framework has been standardised by ISO (International Organisation for Standardisation) and ITU (International Telecommunication Union) [8]. An essential part of this methodology is a notation for the definition of conformance test cases, called TTCN (Tree and Tabular Combined Notation) [9]. TTCN has been designed for protocols and systems for which in general timing between communicating entities is not an issue. Test cases are specified as sequences of test events. The relative ordering of test events is defined in a *behaviour description*.

For some years, the situation and thus the requirements for testing are changing. Two main new kinds of distributed services and systems can be identified: real-time and multimedia systems. Real-time systems stem from the use of computers for controlling physical devices and processes. For these systems, real-time communication is essential for their correct behaviour. Multimedia systems on the other hand involve the transmission of several continuous streams (of bits) between source and sink and their timely reproduction at the sink (e.g., synchronisation of audio and video). As pointed out in, e.g., [1], TTCN is not an appropriate test notation for testing real-time and multimedia systems: Firstly, test events in TTCN are meant to be used for message-based systems and not for stream-based systems. Secondly, in TTCN, real-time can only be approximated but not specified exactly. In this paper we define a real-time extension of TTCN as a contribution for solving the latter problem.

Our extension of TTCN to *real-time TTCN* is on a syntactical and a semantical level. In particular, the syntactical difference is that for real-time TTCN we allow an annotation of test

events with an earliest execution (EET) and a latest execution time (LET). Informally, a test event may be executed if it has been continuously enabled for at least EET units and it must be executed if it has been continuously enabled for LET units. Test events are executed instantaneously. Our approach is similar to [2, 3, 4, 7, 11, 12, 14, 17].

The paper proceeds as follows: The next two sections introduce TTCN and real-time TTCN, respectively. A discussion of the features of real-time TTCN is done in Section 4, where real-time TTCN is used for the definition of test cases for the generic cell rate algorithm used for traffic control in ATM networks. We conclude the paper with a characterisation of real-time TTCN and an outlook.

2 TTCN - Tree and Tabular Combined Notation

TTCN is a notation for the description of test cases to be used in conformance testing. For the purpose of this paper we restrict our attention to TTCN concepts related to the description of the dynamic test case behaviour. Further details on TTCN can be found in [9, 10, 13, 16, 18].

2.1 Abstract Testing Methods and TTCN

A test case specifies which outputs from an *implementation under test* (IUT) can be observed and which inputs to an IUT can be controlled. Inputs and outputs are either *abstract service primitives* (ASPs) or *protocol data units* (PDUs). In general, several concurrently running distributed *test components* (TC) participate in the execution of a test case. TCs are interconnected by *coordination points* (CPs) through which they asynchronously exchange *coordination messages* (CMs). TCs and IUT logically communicate by exchanging PDUs which are embedded in ASPs exchanged at *points of control and observation* (PCOs), which are interfaces above and below the IUT. Since in most cases the lower boundary of an IUT does not provide adequate PCO interfaces, TCs and IUT communicate by using services of an underlying service provider.

2.2 Test Case Dynamic Behaviour Descriptions

The behaviour description of a TC consists of *statements* and *verdict assignments*. A verdict assignment is a statement concerning the conformance of an IUT with respect to the sequence of events that have been performed. A PASS verdict is assigned if the IUT passes the test, FAIL is given if the IUT contradicts the specification, and INCONCLUSIVE is assigned if neither a PASS nor a FAIL verdict can be assigned. TTCN statements are *test events* (SEND, IMPLICIT SEND, RECEIVE, OTHERWISE, TIMEOUT and DONE), *constructs* (CREATE, ATTACH, ACTIVATE, RETURN, GOTO and REPEAT) and *pseudo events* (qualifiers, timer operations and assignments).

Statements can be grouped into *statement sequences* and *sets of alternatives*. In the graphical form of TTCN, sequences of statements are represented one after the other on separate lines and being *indented* from left to right. The statements on lines 1 - 6 in Fig. 1 are a statement sequence. Statements on the same level of indentation and with the same predecessor are alternatives. In Fig. 2 the statements on lines 4 and 6 form a set of alternatives: they are on the same level of indentation and have the statement on line 3 as their common predecessor.

2.3 Test Component Execution

A TC starts the execution of a behaviour description with the first *level of indentation* (line 1 in Fig. 1), and proceeds towards the last level of indentation (line 6 in Fig. 1). Only one alternative out of a set of alternatives is executed, and test case execution proceeds with the next level of indentation relative to the executed alternative. For example, in Fig. 2 the statements on line 4 and line 6 are alternatives. If the statement on line 4 is executed, processing continues with the statement on line 5. Execution of a behaviour description stops if the last level of indentation has been visited, a test verdict has been assigned, or a test case error has occurred.

Before a set of alternatives is evaluated, a *snapshot* is taken [9], i.e., the state of the TC and the state of all PCOs, CPs and expired timer lists related to the TC are updated and frozen until the set of alternatives has been evaluated. This guarantees that evaluation of a set of alternatives is an *atomic* and *deterministic action*.

Test Case Dynamic Behaviour					
Nr	**Label**	**Behaviour Description**	**CRef**	**V**	**Comments**
1		CP ? CM	connected		RECEIVE
2		(NumOfSends := 0)			Assignment
3		REPEAT SendData			Construct
		UNTIL [NumOfSends > MAX]			
4		START Timer			Timer Operation
5		?TIMEOUT timer			TIMEOUT
6		L ! N-DATA request	data		SEND

Figure 1: TTCN Behaviour Description - Sequence of Statements

Test Case Dynamic Behaviour					
Nr	**Label**	**Behaviour Description**	**CRef**	**V**	**Comments**
1		[TRUE]			Qualifier
2	L1	(NumOfSends := NumOfSends + 1)			
3		+SendData			ATTACH
4		[NOT NumOfSends > MAX]			Alternative 1
5		-> L1			GOTO
6		[NumOfSends > MAX]			Alternative 2

Figure 2: TTCN Behaviour Description - Set of Alternatives

Alternatives are evaluated in sequence, and the first alternative which is *evaluated successfully* (i.e., all conditions of that alternative are fulfilled [9]) is executed. Execution then proceeds with the set of alternatives on the next level of indentation. If no alternative can be evaluated successfully, a new snapshot is taken and evaluation of the set of alternatives is started again.

3 Real-Time TTCN

Real-time TTCN extends TTCN on a syntactical and a semantical level (see [22] for a preliminary version of real-time TTCN). Syntactically, in real-time TTCN we allow an annotation of test events with an earliest execution time (EET) and a latest execution time (LET). To be more precise, our extensions of TTCN comprise the definition of a table for the specification of time names and time units and the addition of two columns in the behaviour description tables for the annotation of TTCN statements. Informally, a test event may be executed if it has been continuously enabled for at least EET units and it must be executed if it has been continuously enabled for LET units. Test events are executed instantaneously.

In [22], we have defined an operational semantics based on timed transition systems [6]. To emphasise the similarities of TTCN and real-time TTCN we follow an alternative approach for a definition of semantics, i.e., we propose a refined snapshot semantics which takes time annotations of TTCN statements into account.

3.1 Syntax of Real-Time TTCN

In real-time TTCN, timing information is added in the declarations and the dynamic part of a test suite.

As shown in Fig. 3 he specification of time names, time values and units is done in an Execution Time Declarations table[1]. Time names are declared in the Time Name column. Their values and the corresponding time units are specified on the same line in the Value and Unit columns. The declaration of time values and time units is optional.

EET and LET[2] are predefined time names with default values 0 and infinity. Default time values can be overwritten (Fig. 3). If time values are not specified the default time values 0 and infinity apply for EET and LET.

Besides the static declarations of time values in an Execution Time Declarations table, changing these values within a behaviour description table can be done by means of assignments (Fig. 4).

[1] Apart from the headings the table looks much like the TTCN Timer Declarations table.

[2] We use different font types for distinguishing between syntax, EET and LET, and semantics, EET and LET.

Execution Time Declarations			
Time Name	**Value**	**Unit**	**Comments**
EET	1	s	EET value
LET	1	min	LET value
WFN	5	ms	Wait For Nothing
NoDur		min	No specified value

Figure 3: Execution Time Declarations Table

Test Case Dynamic Behaviour							
Nr	**Label**	**Time**	**TOptions**	**Behaviour Description**	**C**	**V**	**Comments**
1	L1	2, 4	M	A ? DATA request			Time label Mandatory *EET*
2				(NoDur := 3)			Time assignment
3		2, NoDur		A ! DATA ack			
4				(LET := 50)			LET update (ms)
5				A ? Data request			
6		L1 + WFN, L1 + LET	M, N	B ? Alarm			Mandatory *EET* not pre-emptive

Figure 4: Adding EET and LET values to behaviour lines

However, evaluation of time labels should alway result in EET and LET values for which $0 \leq EET \leq LET$ holds. As indicated in Fig. 4 we add a Time and a Time Options column to Test Case Dynamic Behaviour tables (and similar for Default Dynamic Behaviour and Test Step Dynamic Behaviour tables). An entry in the Time column specifies EET and LET for the corresponding TTCN statement. Entries may be constants (e.g., line 1 in Fig. 4), time names (e.g., the use of NoDur on line 3), and expressions (e.g., line 6).

In general, EET and LET values are interpreted relative to the enabling time of alternatives at a level of indentation, i.e., the time when the level of indentation becomes the current level. However, some applications may require to define EET and LET values relative to the execution of an earlier test event, i.e, not restricted just to the previous one. In support of this requirement, a label in the Label column may not only be used in a GOTO but can also be used in the Time column, so that EET and LET values are computed relative to the execution time of the alternative identified by the label: In Fig. 4 on line 6 the time labels (L1 + WFN, L1 + LET) are referring to the execution time of the alternative in line 1 (for which label L1 is defined).

Entries in the Time Options column are combinations of symbols M and N. Similar to using labels in expressions, time option N allows to express time values relative to the alternative's own enabling time even though some TTCN statements being executed in between two successive visits of the same level of indentation.

Thus, the amount of time needed to execute the sequence of TTCN statements in between two successive visits is compensated. If time option N is defined, then execution of this alternative is not pre-emptive with respect to the timing of all alternatives at the same level of indentation.

In some executions of a test case, a RECEIVE or OTHERWISE event may be evaluated successfully before it has been enabled for EET units. If it is intended to define EET as a mandatory lower bound when an alternative may be evaluated successfully, then time option M has to be specified. Informally, if time option M is specified and the corresponding alternative can be successfully evaluated before it has been enabled for EET units, then this results in a FAIL verdict.

3.2 Snapshot Semantics of Real-Time TTCN

In this section we develop a snapshot semantics for RT-TTCN (for more details see also [22]). We assume that the current level of indentation has been expanded as defined in Annex B of [9] and that its general form is $A_1[eexp_1, lexp_1], \ldots, A_n[eexp_n, lexp_n]$, where A_i denotes an alternative and $eexp_i$, $lexp_i$ are expressions for determining EET and LET values. The evaluation of expressions $eexp_i$ and $lexp_i$ depends on whether $eexp_i$ and $lexp_i$ make use of a label Ln. If so, absolute time references are converted into time references relative to the enabling time of the current set of alternatives.

We define the following functions used in the semantics definition. Let *eval* be a function from time expressions to time values for EET and LET. Let $enablingTime(A_i)$ be a function that returns the time when alternative A_i has been enabled. Notice that for all A_i $enablingTime(A_i)$ is the same. Let $executionTime$(Ln) be a function that returns the execution time of an alternative at the level of indentation are referred to by label Ln. And let NOW be a function that returns the current global time. For the evaluation of time expressions the following rules apply:

1. If $eexp_i$ and $lexp_i$ do not involve any operator Ln, then $EET = eval(eexp_i)$ and $LET = eval(lexp_i)$. It is required that $0 \leq EET \leq LET$ holds.

2. If $eexp_i$ and $lexp_i$ involve an Ln then, firstly, $executionTime$(Ln) is substituted for Ln in $eexp_i$ and $lexp_i$ resulting in expressions $eexp'_i$ or $lexp'_i$, and secondly, $EET = eval(eexp'_i) - \text{NOW}$ and $LET = eval(lexp'_i) - \text{NOW}$. It is required that $0 \leq EET \leq LET$ holds. Notice that absolute time references $eexp_i$ and $lexp_i$ are converted into time references EET and LET relative to the current global time NOW.

We say that alternative A_i is *potentially enabled* if A_i is in the current set of alternatives. A_i is *enabled* if A_i is evaluated successfully (Sect. 2.3), A_i is *executable* if A_i is enabled and A_i has been potentially enabled for at least EET_i and at most LET_i time units. The refined snapshot semantics of real-time TTCN is defined as follows:

1. (a) If the level of indentation is reached from a preceding alternative (not by a GOTO or RETURN), then all alternatives are marked *potentially enabled* and the global time is taken and stored. It is accessible by $enablingTime(A_i)$.

   (b) If the level of indentation is reached by executing a GOTO or RETURN and enablingTime(A_i) has been frozen earlier, then all alternatives are marked *potentially enabled* but $enablingTime(A_i)$ is not updated.

   (c) If the level of indentation is reached by executing a GOTO or RETURN but enablingTime(A_i) has not been frozen earlier, then all alternatives are marked *potentially enabled* and the global time is taken and stored. It is accessible by $enablingTime(A_i)$.

   (d) Otherwise, the steps 1- 4 are iterated again.

   EET and LET are computed as described above. PCO, CP and expired timer lists are locked. If for an A_i $enablingTime(A_i) + LET_i < \text{NOW}$, then test case execution stops with a FAIL verdict.

2. All alternatives which can be evaluated successfully are marked *enabled*. If for an enabled alternative, say A_i, time option M is set and if $\text{NOW} < enablingTime(A_i) + EET_i$, then test case execution stops with a FAIL verdict. If no alternative in the set of alternatives can be evaluated successfully, then PCO, CP and timer lists are unlocked and processing continues with Step 1.

3. An enabled alternative A_i is marked *executable* provided that $enablingTime(A_i) + EET_i \leq \text{NOW} \leq enablingTime(A_i) + LET_i$ and if there is another enabled alternative A_j with $enablingTime(A_j) + EET_j \leq \text{NOW} \leq enablingTime(A_j) + LET_j$, then $i < j$, i.e., the i-th alternative precedes the j-th alternative in the set of alternatives. If no alternative can be marked executable, then PCO, CP and timer lists are unlocked and processing continues with Step 1.

4. An executable alternative A_i is executed. If a label Ln is specified, then the execution time of the alternative is recorded (and can be accessed by $executionTime$(Ln)). If for the executed alternative time option N is specified $enablingTime(A_i)$ is frozen. PCO, CP and timer lists are unlocked. The next level of indentation is visited.

Remarks: If any potentially enabled alternative cannot be evaluated successfully before LET, then a specified real-time constraint has not been met and test case execution stops with a FAIL verdict. Conversely, if an alternative can be evaluated successfully before it has been potentially enabled for EET time units (Step 2), then a defined real-time constraints is violated, too. In Step 3, the selection of alternatives for execution from the set of enabled alternatives follows the same rules as in TTCN [9].

Test Case Dynamic Behaviour							
Nr	**Label**	**Time**	**Time Options**	**Behaviour Description**	**CRef**	**V**	**C**
1	L1	2, 4	M	PCO1 ? N-DATA indication	info		
2				...			
3				-> L1			
4		0, INFINITY		PCO2 ? N-ABORT indication	abort		
5				...			

Figure 5: Partial Real-Time TTCN Behaviour Description

Test Case Dynamic Behaviour					
Nr	**Label**	**Behaviour Description**	**CRef**	**V**	**C**
1	L1	START T1(EET)			
2		?TIMEOUT T1 START T2(LET-EET)			
3		PCO1 ? N-DATA indication	data		
4		-> L1			
5		PCO2 ? N-ABORT indication STOP T2	abort	INCONC	
6		?TIMEOUT T2		FAIL	
7		PCO2 ? N-ABORT indication STOP T1	abort	INCONC	
8		PCO1 ? OTHERWISE STOP T1		FAIL	

Figure 6: TTCN Behaviour Description for the ISDN Example

4 Application of Real-Time TTCN — ISDN and Generic Cell Rate Algorithm

In ISDN (Integrated Services Digital Network) [5], the B-channels are used by applications for data exchange whereas the D-channel is used for the management of connections between users or application processes. We consider a scenario where an ISDN connection between test system and IUT has been established and where PCO1 and PCO2 are the respective B- and D-channel interfaces. At the B channels we expect to receive user data every $EET_1 = 2$ to $LET_1 = 4$ time units. At any time the ISDN connection may be aborted on the D channel.

We consider the partial real-time TTCN behaviour description given in Fig. 5. The first alternative A_1 may be evaluated successfully and may be executed only in the interval $EET_1 = 2$ and $LET_1 = 4$ because time option M is set on line 1. Let us assume that at T' with enablingTime(A_1)+ $EET_1 \leq T' \leq$ enablingTime$(A_1) + LET_1$, an N-DATA indication is received. The first alternative may be executed at T'' with enablingTime$(A_1) + EET_1 \leq T' \leq T'' \leq$ enablingTime$(A_1) + LET_1$ (Step 4) because no other alternative is executable (no N-ABORT indication has been received yet). Suppose that an N-DATA indication and an N-ABORT indication have been received at $T''' : T' \leq T''' \leq T''$. Then, although both alternatives are executable, the first alternative is executed because of the ordering of alternatives in the set of alternatives (Step 3). If an N-DATA indication is received at $T <$ enablingTime$(A_i) + EET_1$, then test case execution stops with a FAIL verdict (Step 2). If no N-DATA indication and no N-ABORT indication have been received before LET_1 time units after the alternatives have been potentially enabled, test case execution stops with a FAIL verdict (Step 1).

In Fig. 6, a test case in TTCN is given for the ISDN example. The timing constraints on the reception of N-DATA indications are expressed using timers T1 and T2. The alternatives coded on lines 2 and 8 in combination check that an N-DATA indication should not be received before EET (= timer T1); otherwise, test case execution results in a FAIL verdict (line 8). The TIMEOUT event on line 6 controls the latest execution time, and if timer T2 expires, then this gives a FAIL verdict.

Let us assume that test case execution is at the third level of indentation (lines 3, 5 and 6) and that TIMEOUT of timer T2 precedes reception of an N-DATA indication. Furthermore, let us assume that the system executing the test case is heavily loaded and therefore evaluation of a set of alternatives lasts too long, so that both events are included in the same snapshot. The late arrival of an N-DATA indication gets undetected because of the ordering of alternatives on line 3, 5 and 6. This problem can be fixed if lines 3 and 6 are exchanged. But now let us assume that an N-DATA indication is received before TIMEOUT of timer T2. A fast system will take a snapshot

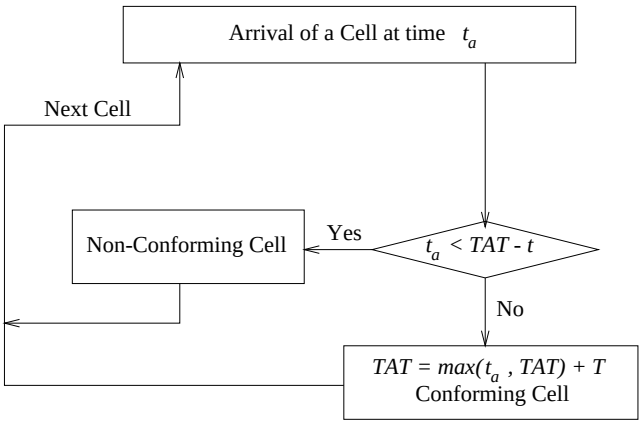

Figure 7: Generic Cell Rate Algorithm (GCRA) - Virtual Scheduling [15]

which includes the N-DATA indication only, whereas a slow system will take a snapshot which includes an N-DATA indication and a TIMEOUT. If the latter happens, the TIMEOUT succeeds over the RECEIVE event. Unfortunately, the behaviour description does not comply with the requirement stated in [8] "that the relative speed of the systems executing the test case should not have an impact on the test result" and thus is not valid.

In ATM (Asynchronous Transfer Mode) networks [15], network traffic control is performed to protect network and users, and to achieve predefined network performance objectives. During connection set up a *traffic contract specification* is negotiated and agreed upon between users and network. A contract specification consists of the connection traffic descriptor, the requested quality-of-service class and the definition of a compliant connection.

A connection is termed *compliant* as long as the number of non-conforming cells does not exceed a threshold value negotiated and agreed upon in the traffic contract. If the number of non-conforming cells exceeds the threshold, then the network may abort the connection. The procedure that determines conforming and non-conforming cells is known as the *generic cell rate algorithm* (GCRA, Fig. 7) [15]: The algorithm calculates the theoretically predicted arrival times (TAT) of cells assuming equally spaced cells when the source is active. The spacing between cells is determined by the minimum inter-arrival time T between cells, which computes to $T = 1/R_p$ with R_p being the peak cell rate (per seconds) negotiated for the connection. If the actual arrival time of a cell t_a is after TAT $- \tau$, τ the cell delay variation tolerance caused, for instance, by physical layer overhead, then the cell is a conforming cell; otherwise, the cell is arriving too early and thus is being considered as a non-conforming cell.

A possible test purpose derivable from the informal definition of traffic contract specification and GCRA may be as follows: "It is to be tested that the amount of traffic (in terms of ATM cells) generated at the UNI is compliant to the traffic contract specification".

For the following discussion we assume a testing scenario as depicted in Fig. 8. The IUT, i.e., the end-system of the user, is connected to an ATM switch which in this scenario is the test system. Several traffic sources may generate a continuous stream of ATM cells which is, by the virtual shaper, transformed into a cell stream compliant with the traffic contract. It is the test system that checks compliance of the received cell stream to the traffic contract.

The definition of a test case assumes that a connection has already been established, so that a traffic contract specification is available. From the traffic contract, parameters R_p, T and τ can be extracted which are assigned to test case variables. The threshold value (for determining when a connection is to be aborted) is provided as a test suite parameter. For simplicity we let $\tau = 0$.

The definition of the dynamic test case behaviour (Fig. 9) is based on the observation that according to the GCRA (Fig. 7), except for the first cell, at most every $T(= EET)$ time units an ATM cell is expected from the IUT. Since we do not expect an ATM cell to arrive before T time units, time option M is defined. If an ATM cell arrives before T time units, then the test case is terminated with a FAIL verdict.

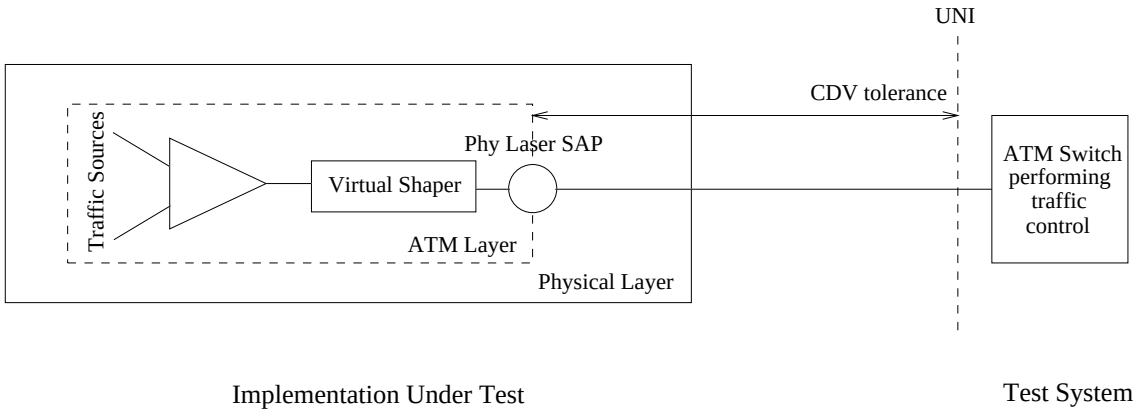

Figure 8: Generic Cell Rate Algorithm - Testing Scenario [15]

Test Case Dynamic Behaviour							
Nr	**L**	**Time**	**T Options**	**Behaviour Description**	**C**	**V**	**C**
1		0, INFINITY		UNI ? ATM-Cell	*		First cell to initialise GCRA
2	L2	T, INFINITY	M	UNI ? ATM-Cell	*		
3				-> L2			

Figure 9: Real-Time TTCN Behaviour Description for GCRA - Threshold = 0

This test case implies a threshold value of zero. If we allow for a number of non-conforming cells (NCC) greater than zero, then the test case definition is as shown in Fig. 10. The difference compared to the previously discussed test case is that whenever an ATM cell arrives before T time units, then counter NCC is incremented and is checked against the defined threshold. Time option N on line 2 instructs the system not to pre-empt the time constraint of the current set of alternatives. If control returns to level L2 from line 5, the enabling time is not updated.

We have shown the use of time labels and time options. Without time options the specification of both test cases would have been more complex. For the first test case it would have been necessary to introduce a second alternative similar to line 2 of Fig. 10 instead of using time option M. For the second test case without time option N calculations of absolute and relative time values would have be necessary in order to adjust EET. Nonetheless, without real-time features, test case specification for both scenarios would have not been possible.

Test Case Dynamic Behaviour							
Nr	**L**	**Time**	**TOptions**	**Behaviour Description**	**C**	**V**	**C**
1		0, INFINITY		UNI ? ATM-Cell (NCC := 0)	*		
2	L2	0, T	N	UNI ? ATM-Cell (NCC := NCC + 1)	*		
3		0, 0		[NCC > Threshold]		FAIL	
4		0, 0		[NCC <= Threshold]			
5				-> L2			
6		T, INFINITY		UNI ? ATM-Cell	*		
7				-> L2			

Figure 10: Real-Time TTCN Behaviour Description for GCRA - Threshold > 0

5 Conclusions and Outlook

We have defined syntax and semantics of real-time TTCN. On a syntactical level, TTCN statements can be annotated by time labels. Time labels are interpreted as earliest and latest execution times of TTCN statements relative to the enabling time of the TTCN statement. The operational semantics of real-time TTCN has been defined in terms of a refined snapshot semantics, thereby

emphasising the close relationship of TTCN and real-time TTCN. If we assume that no time values are defined (in this case EET and LET are zero and infinity, respectively), execution of a test case results in the same sequence of state-transitions as in TTCN. Therefore, our definition of real-time TTCN is compatible to TTCN.

Real-time TTCN combines property and requirement oriented specification styles. Time labels for TTCN statements, in general, define real-time constraints for the test system. A test system should be implemented so that it can comply with all properties defined. Time labels for RECEIVE and OTHERWISE events, which imply a communication with the IUT, define requirements on the IUT and the underlying service provider. For real-time TTCN, the underlying service provider should also be sufficiently fast with respect to the timing of activities. Therefore, if a timing constraint of a RECEIVE or OTHERWISE event is violated, this clearly is an indication that the IUT is faulty and the test run should end with a FAIL verdict assignment.

Our future work will focus on the definition of appropriate test architectures for testing real-time requirements and on the development of editors, compiler and run-time libraries for real-time TTCN. A possible theoretical extension of our approach is to allow the use of time labels at a more detailed level, e.g., the annotation of test events, assignments and timer operations (an extension of [20, 21]).

Acknowledgements. The authors are indebted to Stefan Heymer for proofreading and for his detailed comments on earlier drafts of this paper.

References

[1] A. Ates, B. Sarikaya. *Test sequence generation and timed testing.* In Computer Networks and ISDN Systems, Vol. 29, 1996.

[2] H. Bowman, L. Blair, G. Blair, A. Chetwynd. *A Formal Description Technique Supporting Expression of Quality of Service and Media Synchronization.* In Multimedia Transport and Teleservices. LNCS 882, 1994.

[3] B. Berthomieu, M. Diaz. *Modeling and Verification of Time Dependent Systems Using Time Petri Nets.* In IEEE Transactions on Software Engineering, Vol. 17, No. 3, March 1991.

[4] S. Fischer. *Implementation of multimedia systems based on a real-time extension of Estelle.* In Formal Description Techniques IX Theory, application and tools, 1996.

[5] F. Halsall, *Data Communications, Computer Networks and Open Systems*, Addison-Wesley, 1994.

[6] T. Henzinger, Z. Manna, A. Pnueli. *Timed Transition Systems.* In Real-Time: Theory in Practice. LNCS 600, 1991.

[7] D. Hogrefe, S. Leue. *Specifying Real-Time Requirements for Communication Protocols.* Technical Report IAM 92-015, University of Berne, 1992.

[8] ISO/IEC. *Information Technology - OSI - Conformance Testing Methodology and Framework - Part 1: General Concepts.* ISO/IEC IS 9646-1, 1994.

[9] ISO/IEC. *Information Technology - OSI - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN).* ISO/IEC IS 9646-3, 1996.

[10] F. Kristoffersen, T. Walter. *TTCN: Towards a formal semantics and validation of test suites.* In Computer Network and ISDN Systems, Vol. 29, 1996.

[11] L. Léonard, G. Leduc. *An Enhanced Version of Timed LOTOS and its Application to a Case Study.* In Formal Description Techniques VI, North-Holland, 1994.

[12] S. Leue. *Specifying Real-Time Requirements for SDL Specifications - A Temporal Logic Based Approach.* In Protocol Specification, Testing and Verification XV, 1995.

[13] R. Linn. *Conformance Evaluation Methodology and Protocol Testing.* In IEEE Journal on Selected Areas in Communications, Vol. 7, No. 7, 1989.

[14] P. Merlin, D. Faber. *Recoverability of Communication Protocols.* In IEEE Transactions on Communication, Vol. 24, No. 9, September 1976.

[15] M. Prycker. *Asynchronous transfer mode: solutions for broadband ISDN*, 3rd Edition, Prentice Hall, 1995.

[16] R. Probert, O. Monkewich. *TTCN: The International Notation for Specifying Tests of Communications Systems.* In Computer Networks and ISDN Systems, Vol. 23, 1992.

[17] J. Quemada, A. Fernandez. *Introduction of Quantitative Relative Time into LOTOS.* In Protocol Specification, Testing and Verification VII, North-Holland, 1987.

[18] B. Sarikaya. *Conformance Testing: Architectures and Test Sequences.* In Computer Networks and ISDN Systems, Vol. 17, 1989.

[19] I. Sommerville. *Software engineering.* 3rd Edition, Addison-Wesley, 1989.

[20] T. Walter, J. Ellsberger, F. Kristoffersen, P.v.d. Merkhof. *A Common Semantics Representation for SDL and TTCN.* In Protocol Specification, Testing and Verification XII, North-Holland, 1992.

[21] T. Walter, B. Plattner. *An Operational Semantics for Concurrent TTCN.* In Proceedings Protocol Test Systems V, North-Holland, 1992.

[22] T. Walter, J. Grabowski. *A Proposal for a Real-Time Extension for TTCN.* In Kommunikation in Verteilten Systemen '97, Informatik aktuell, Springer Verlag, 1997.