Evolution of Real-Time TTCN Testing based on Prioritised Scheduling

C.Kavadias*, A.Litke*, G.Thanos*, V.Kollias* and W.Skelton**

*TELETEL S.A; 124, Kifisias Avenue, Athens, GREECE
Tel: + 30 1 6983393; Fax: +30 1 6983391;

**SOLINET GmbH Solutions for Innovative Networks
Mittlerer Pfad 26, 70499 Stuttgart, Germany
Tel +49 711 1398 13-0, Fax +49 711 866 1240

E-mail: C.Kavadias@teletel.gr

Abstract: Work in this paper identifies specific problems when TTCN is used for real-time testing together with ways these problems can be avoided or their effect at least minimised. This contribution proposes the main real-time extensions that are necessary to be supported by the TTCN language for real-time testing, specifically the prioritised timers and the prioritised coordination messages extensions. Finally, it proposes how the current timer and coordination message syntax should be extended in order to introduce the prioritised timers and the prioritised coordination messages concept.

Keywords: TTCN, real-time testing, prioritised timers, prioritised coordination messages

1. Introduction

Modern telecommunication systems, such as mobile phone systems, belong to the class of distributed real-time systems. The real-time aspects of these systems become increasingly important, because of the communication of multimedia contents (e.g. in UMTS), which put tough requirements on the performance aspect. In general, all telecommunication systems have functional requirements (e.g. provided service) and non-functional requirements (e.g. performances, response time). Functional aspects of the system can only be fully validated and tested if the non-functional, especially timed aspects are satisfied. Current formal techniques for the design and validation of such systems, e.g. SDL and TTCN, cover only parts of timed aspects.

On the other hand, there exist already highly professional and efficient tool environments for SDL, MSC and TTCN, which integrate all aspects of specification, design, validation, implementation and testing. Most of the major telecommunication equipment manufacturers and network operators use these environments in their system development processes. However, so far the applicability of the tools is constrained by the missing real-time functionality.

2. Background

Both TTCN-2 [ISO-9646-3] and TTCN-3 [ITU recommendation Z.140] are powerful enough for describing conformance test suites. The available timer constructs can be used to some extent for real-time test requirements, but only by accepting the implicit
latency for timeouts, which causes specific problems when TTCN is used for real-time testing. This article identifies these problems together with ways these problems can be avoided or their effect at least minimised.

The simple timer concept intended to catch long-term timeouts is not powerful enough to deal with hard real-time. Since the standard snapshot semantics may summarize time-critical events into one snapshot, important timing or ordering information might get lost. This is in contrast to the TTCN standard which states “a Test Case or Test Step should not contain behaviour where the relative processing speed of the MOT (Means of Testing) could impact the results.” [ISO 9646-3]. In this case, it is not decidable, whether a violation of real-time constraints occurred or not. The verdict will rather depend on the question of how the alternatives are ordered in the TTCN dynamic behavior description. Therefore [ISO 9646-3] suggest “[...] to avoid unreliable results caused by situations in which the order of receipt of events at different PCOs or CPs is used to determine verdict assignment”. But exactly such behaviour is needed to describe real-time test cases.

Additionally, a tester may consist of many Parallel Test Components (PTCs) and a main test component (MTC). In order to be able to test for timed behaviour spread over different PTCs, absolute priority to the co-ordination messages between the PTCs is required. This means that the synchronisation of the PTCs requires a solution where co-ordination messages have increased priority and are served interruptively by the scheduler of the simulation engine.

Following the investigation of these requirements, it has been concluded that the main real-time extensions what is necessary to be supported by the TTCN simulator is the prioritised timers and the prioritised co-ordination messages concepts. Although TTCN provides a timer mechanism which allow to set timers and to check their status, the absolute and relative timing of events cannot be specified in both TTCN (TTCN-2, TTCN-3) versions. The speed of the tester can influence the validation results of the IUT in case of timed critical functional testing. To overcome this drawback the prioritised timers is proposed as a real-time extension to the existing TTCN languages. The prioritised timer extension is applicable to both TTCN-v2 and TTCN-v3 can be incorporated in both as well.

3. Prioritised scheduling

The study that follows is focusing on system behaviour in overload conditions. Overload conditions are exceptional and the effect on test case behaviour is implementation and platform specific.
The issue of scheduling is not explicitly addressed in TTCN, which works instead with a snapshot concept of events pending at the PCOs. Under normal circumstances scheduling issues are not relevant for this mechanism, but under overload conditions this is no longer true.

Before events (not the expired timers) are added to the snapshot they have to pass through a platform specific run-time mechanism (execution engine), as depicted in Figure 1. This is the case for any simulation platform and the run-time behaviour will depend on the platform and the implementation algorithms used. This is the point at which the TTCN extensions are relevant and through which the run-time behaviour can be optimised.

The prioritised timers and the prioritised co-ordination messages extensions are a generic mechanism to guide a TTCN compiler to optimise the run-time behaviour under overload conditions. It is not a contradiction to any of the existing specifications, but a generic mechanism that can be optionally applied. This way a developer of TTCN test suites has the ability to control the execution of the test cases whether the tester is under overload conditions or not.

4. Prioritised Timers

From the TTCN (ISO 9646-3) reference architecture, there are PCOs distributed over the instanced test components; each PCO has a queue to store the arriving events. Events are processed in the order they arrive (for unblocked PCOs). A PCO can be blocked if the first event in its queue doesn't match any of the options in the corresponding test component. When the test component moves to new options, for example when a matching event arrives at a different PCO, then the events pending at blocked PCOs are checked again and may be consumed. A timeout is handled immediately, is not associated with any PCO and overtakes all other events.

An implementation of an executable test scenario runs on a host platform and is subject to the real constraints associated with the physical characteristics of that platform, including performance. This means that events can arrive faster than they are processed, at least for a limited period of time (burst traffic). These events have to be stored in an event queue, which is implicit in the ISO 9646-3 specification, and follows from the reality of 'bursty' traffic.
Considering the behaviour of a PCO, an empty PCO is, at least for the first event, fully transparent. The event is not delayed (or stored) in the PCO unless there were no matching options in the corresponding test component. Furthermore, if the event is stored in the PCO, it means the PCO is blocked and hence excluded from the normal scheduling. This means that from an implementation point of view the effective scheduling behaviour of (non-timeout) events in TTCN test scenarios is fully equivalent to the SDL scheduling model, at least for unblocked PCOs, and furthermore that this is fully according to the ISO 9646-3 reference architecture.

Considering the other classes of events, namely for blocked PCOs and timeouts, these are also fully compatible with the SDL scheduling model within the reality of the assigning processor time in the host platform, again according to ISO 9646-3 under 'bursty' traffic conditions.

For example, in the case of blocked PCOs, the only condition for a PCO to become unblocked is for a matching event to arrive at a different PCO allowing the test component to move to new options. At this point the events pending at the blocked PCOs have to be re-checked as if they had just arrived. The allocation of processor time to each pending event can correspond strictly and exactly to the same algorithm, that is used to queue events that arrive faster than they can be processed (see burst traffic above).

Of course a different algorithm could also be used, but this is not necessary for further analysis of the scheduling as the same algorithm can be used. Therefore it follows that the scheduling of non-timeout events at PCOs is in all cases equivalent to the scheduling of events arriving faster than they can be processed.

Similarly, considering the case of a timeout occurring in a test component, there is the case that when there are no pending events (in the burst traffic situation) then the timeout will be the next event to be processed. In the case of pending events, the timeout has to overtake the pending events, in effect being added to the front of the event queue.

Whilst this is not according to the SDL scheduling model, in the case that the timer priority 'Add to End of Queue' is used, it follows that the scheduling is then fully according to the SDL scheduling model.

Likewise if the concept of prioritised timers is applied to the SDL scheduling model, it follows that the mode 'Add to Front of Queue', if applied to SDL timers, corresponds exactly to the TTCN scheduling model.

This conclusion confirms that the TTCN and SDL scheduling models are in fact, with the proposed extensions, fully equivalent and subject to the same limitations and constraints of physical platforms operating in an overload condition, which is exactly the situation addressed in this study. It also extends the completeness of design to cover the entire TTCN and SDL specifications, which is, at least from a software engineering point of view, a very satisfying result.

Furthermore follows that the use of prioritised timers is a convenient way to control and influence the behaviour of test scenarios running in overload conditions; something that is not possible in the 'vanilla' version of TTCN.
As the scheduling model for TTCN-3 is fully compatible with the above concepts, it follows that the identified TTCN-2 extensions for prioritised timers can be transferred to TTCN-3 at least conceptually, if not syntactically.

Additionally, there are two cases to be examined with the use of the priority timer concept concerning scheduling mechanisms. There are different approaches for preemptive and non-preemptive scheduling algorithms. Specific requirements raised by the two scheduling algorithms should be further examined in order to draw the exact type of support needed for the priority timer.

4.1. TTCN Timer Extension

According to the ISO/IEC 9646-3 / 1997 reference of TTCN, the following information shall be provided for each timer:

a) the timer name,

b) the optional timer duration,

where the default duration of the timer shall be an expression which may be omitted if the value cannot be established prior to execution of the test suite; the terms in the value expression shall not contain: Test Suite Variables or Test Case Variables; the timer duration shall evaluate to an unsigned positive INTEGER value;

c) the time unit,

where the time unit shall be one of the following:

1) ps (i.e., picosecond);
2) ns (i.e., nanosecond);
3) us (i.e., microsecond);
4) ms (i.e., millisecond);
5) s (i.e., second);
6) min (i.e., minute).

Time units are determined by the test suite designer and are fixed at the time of specification. Different timers may use different units within the same test suite. If a PICS or PIXIT entry exists, the timer declaration shall specify the same units included in the PICS/PIXIT entry.

### Table 1: Timer Declarations

<table>
<thead>
<tr>
<th>Group</th>
<th>Timer Name</th>
<th>Duration</th>
<th>Unit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimerIdentifier</td>
<td>[Declaration Value]</td>
<td></td>
<td>TimeUnit</td>
<td>[Free Text]</td>
</tr>
</tbody>
</table>

Detailed Comments: [Free Text]
Within this article we propose the Timer Declaration syntax to be extended with one additional field introducing the concept of the Prioritised Timer. The difference between the old syntax and the new one is the addition of a new field named **PriorityId**, under the TimerId field. As a consequence the information that shall be provided for each timer under this new concept is the following:

- a) the timer name,
- b) the optional priority identifier,

which can be assigned the following keywords:

1) **Off**: Normal add to queue (could be renamed Normal).
2) **Low**: Stick to back until no other events to process.
3) **Medium**: Go to front, but behind High priority events.
4) **High**: Go to front. Next event to be processed.

- c) the optional timer duration,
- d) the time unit,

As the priority is optional there should be a default value in case it is omitted. So, if no priority is specified then the default (for TTCN standard behaviour) is High.

This distinction gives the opportunity to have four kinds of Timers, making the TTCN tool’s scheduler of the simulation engine to perform special handling for each one of them.

In case that the PriorityId field is set with the keyword Off, corresponding timer events are served in the same order as the signalling events. The special handling is performed in case that Low, Medium or High keywords are set. Timers with low priority (Low keyword is set) cause signalling events to overtake any low priority timer events pending. In the case of Timers with medium priority (Medium keyword is set), the medium priority timer events overtake other signalling events, but not any other pending timer events. Finally, a high priority Timer (High keyword is set) causes all high priority timer events to be served before all other pending events.

So, the information for the new Timer Declaration shall be provided in the format shown in the following table:

<table>
<thead>
<tr>
<th>Group</th>
<th>Timer Name</th>
<th>Priority</th>
<th>Duration</th>
<th>Unit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TimerIdentifier</td>
<td>[PriorityIdentifier]</td>
<td>[Declaration Value]</td>
<td>TimeUnit</td>
<td>[Free Text]</td>
</tr>
</tbody>
</table>

**Table 2: Timer Declarations with Priority Identifier**
5. Prioritised Co-ordination Messages

A tester may consist of many Parallel Test Components (PTCs) and a main test component (MTC). In order to be able to test for timed behaviour spread over different PTCs, absolute priority to the co-ordination messages between the PTCs is required. This means that the synchronisation of the PTCs requires a solution where co-ordination messages have increased priority and are served interruptively by the scheduler of the simulation engine.

Coordination Messages (CM) are very similar to Abstract Service Primitives (ASP) except that they occur at Coordination Points (CP). Either the tabular form or ASN.1 as defined in CCITT Recommendation X.208 [27] can be used.

CMs are treated like ASPs and not as Protocol Data Units (PDUs) because TTCN is not concerned with the encoding of these messages. That is a matter for the implementors of the test suite.

A CM parameter may be of any TTCN type including structured types and the metatype PDU. Generally, though, it is recommended that CMs are kept as simple as possible. In many cases a CM will not even have parameters, the name itself will be adequate (e.g. STOP, WAIT, etc.). There are no predefined TTCN CMs.

The same extensions as in the case of prioritised timers can be also applied to the prioritised co-ordination messages ensuring the simulation engine’s special handling according to their priority.

Within this article, we propose the CM constraint declaration syntax to be extended with one additional field introducing, in the same way as in Prioritised Timers declaration, the Prioritised CM constraint declaration. As in case of Timers, the priority is optional and remains under the same priority scheme in order to combine both events and have a common meaning. So, the PriorityId can take the following four values:

1) Off: Normal add to queue (could be renamed Normal).
2) Low: Stick to back until no other events to process.
3) Medium: Go to front, but behind High priority events.
4) High: Go to front. Next event to be processed.

In case that no priority is specified, then the default for the CM is Off (same priority as other events).

The following table shows the suggested format for the prioritised CM constraint declaration:
6. Validation of the TTCN timed extensions

The TTCN timed extensions have been already validated and tested in complex test scenarios and under overload conditions using real equipment. Major telecom manufacturers have reported problems that occurred during the testing operation under overload conditions, specifically with some procedure of the low layer protocols. The low layer protocols (data link) are of great importance for ensuring the well operation of a system (e.g., switch) and impose many timed critical constraints. Precisely, during the execution of specific test suites against real equipment under overload conditions, the verdicts of these test suites were not the ones expected, meaning that they returned FAIL without a specific problem of the System Under Test (SUT). On the other hand, after the integration of the TTCN timed extensions into their testing environments, they have executed the same test suites against real equipment under overload conditions and the verdicts for the same test cases were PASS.

7. Real-time TTCN behaviour based on prioritised scheduling

Summarising the concepts in the above analysis, the prioritised timers and the prioritised co-ordination messages extensions to the TTCN notation refer to a generic mechanism that guides the TTCN compiler to optimise the run-time behaviour (execution engine). Under normal circumstances scheduling issues are not relevant to the current TTCN snapshot mechanism, but under overload conditions this is no longer true.
Figure 2 TTCN run-time behaviour under overload conditions

Figure 2 depicts the optimised TTCN run-time mechanism including the prioritised timers and the prioritised co-ordination messages concepts, as they are suggested in the present article. Before events and expired times are added to the snapshot mechanism they have to pass through a platform specific run-time mechanism (execution engine). It is not a contradiction to any of the existing specifications, but a generic mechanism that can be optionally applied. This way a developer of TTCN test suites has the ability to control the execution of the test cases when the tester is under overload conditions.