TTCN-3 User Conference 2009
3 – 5 June 2009 - ETSI, Sophia Antipolis, France

Testing Embedded Systems in the Automotive Industry with TTCN-3

Jens Grabowski (Universität Göttingen), Jürgen Großmann (Fraunhofer FOKUS), Friedrich-Wilhelm Schröer (Fraunhofer FIRST), Jacob Wieland (Testing Technologies), Diana Serbanescu (Technische Universität Berlin)

grabowski@informatik.uni-goettingen.de, juergen.grossmann@fokus.fraunhofer.de, f.w.schroeer@first.fraunhofer.de, wieland@testingtech.com, diana.serbanescu@fokus.fraunhofer.de
Test Specification Technology and Methodology for Embedded Real Time Systems in the Automobile

- Testing discrete and continuous real time systems with TTCN-3 embedded.
- Test support for the entire integration process.
- Exchange of test definitions between
  - OEM and supplier
  - various test- and simulation platforms e.g. Model in the Loop (MIL) platforms, Software in the Loop (SIL) platforms, and Hardware in the Loop (HIL) platforms
- Integration with model based development especially with AUTOSAR.
- Analysis and improvements of test quality.
Motivation

- Testing software based embedded systems steadily increase in complexity.
- In addition to that non-functional requirements, especially time related input-output behavior, have to be considered.
- Adequate and standardized test solutions are needed, which at least feature a minimum of flexibility, reusability and abstraction.

---

- **GOAL: Provide a standardized testing solution for standardized development environments (e.g. AUTOSAR for Automotive Solutions).**
- **GOAL: Tight Integration of real time testing concepts in an existing test specification environment (i.e. The Test and Testing Control Notation)**
TTCN-3 embedded Tasks

- TTCN-3 embedded for real time systems
- TTCN-3 embedded for continuous behavior
- TTCN-3 embedded hybrid behavior
  - Graphical presentation format for TTCN-3 embedded
  - Preparation for standardization
Real Time Test System Requirements

- **Standard:** assessment of functional behavior (e.g. message contents).
- **Additional:** exact measurement, comparison and assessment of message timing.
- **Additional:** temporal control of message dispatching.
timer t1,t2;
p_out.send(OUT_MSG_1);
t1.start(t_max);
alt{
    []p_in.receive(IN_MSG_1){setverdict(pass)};
    []t1.timeout{setverdict(fail)}
}
t2.start(twait);
t2.timeout;
p_out.send(OUT_MSG_2);
p_in.receive(IN_MSG_2);
var float r_time,s_time;
p_out.send(OUT_MSG_1);
s_time:=now;
p_in.receive(IN_MSG_1)->timestamp r_time;
if(r_time>s_time+tmax) setverdict(fail);
wait(r_time+twait);
p_out.send(OUT_MSG_2);
p_in.receive(IN_MSG_2);
setverdict(pass);
Formalization of the Test System

\[ TS = \{P, Q, C, M, TP, OP\} \]

- a set \( P \) of ports to communicate with the System Under Test (SUT),
- a set \( Q \) of input queues to organize the order of incoming messages,
- a set \( C \) of synchronized clocks to measure time and to simulate TTCN-3 timers,
- a set \( M \) of messages,
- a set \( TP \subseteq TP_{data} \cup TP_{time} \) of predicates that are used to characterize the properties of incoming messages, and
- a set \( OP = \{\text{snap, check, enqueue, dequeue, first, encode, decode, match}\} \) of time-consuming operations that are necessary to organize the handling of messages at ports.
Temporal estimations are only possible on basis of the assumption $t_{receive_{mh}} \approx t(c^s)$, i.e. the time point of taking the snapshot approximates the reception time of messages.
Example: Comparison of Message Timing in Standard TTCN-3

- Arrival of messages $m_0, \ldots, m_n$ and the timeout of timers $t_0, \ldots, t_m$ are denoted by events $e_0, \ldots, e_{k+1}$,
  - timing is measured by comparison of events, and
  - only events that occur in different snapshots are distinguishable.

\[ t(e_0) \leq t(e_1) = t(e_2) = \ldots \leq t(e_{k+1}) \]

- Duration between two consecutive snapshot denote the best accuracy of time measurement for standard TTCN-3. The duration depends on:
  - the number of messages that arrive and the number of ports (queues) to check,
  - the duration of check, decode, match for individual messages where the duration of decode and match is directly dependent on the content and structure of the message under observation.
Problematic Situations: message burst over one or multiple ports.

Each alternative is defined by: $a_k = (q_{a_k}, tp_{d_{a_k}}) \in ALT \subseteq Q_{alt} \times TP_{alt}$

*Simple assumption:* a new message has arrived at each port and none of the messages match.

$$worst(t(s_{n+1}) - t(s_n)) =$$

$$\sum_{x=1}^{l} (dur(check(q_{a_x}^{s_n})) + dur(decode(m_{a_x}^{q_{a_n}})))$$

$$+ dur(match(m_{a_x}^{q_{a_n}}, tp_{a_x})) + dur(snap)$$
Solution

- Seamless access to time
- Explicit measuring and access to the reception time of messages
- Utilities to handle comparison of time and temporal control of statement execution
Time: Concepts & Representation

- Time model based on positive real numbers $t \in \mathbb{R}^+$
- Actual time $t = t(c_0)$ can be directly obtained by the user (now operator).
- TTCN-3 Language Level:
  - **now** operator returns time in seconds coded as a float value.
  - we allow arithmetic expressions on time values
  - precision of time measurement can be specified by means of the **precision** annotation

```plaintext
module{
  ...
  var float myTimeVar;
  testcase myTc runs on myComp{
    ...
    myTimeVar := now + 1.0;
    ...
  }
  with{precision:=0.001}
}
```
... to retrieve the enqueue time of a message,

```tcl
p.receive(t) -> timestamp myTime;
// yields the reception time of a message
```

and time measurement at any place in the test

```tcl
var float myTime := now;
// yields the actual time
```
Verification of Temporal Behaviour

- Verification of enqueue time for incoming messages, procedure calls etc.

```plaintext
p.receive(t) -> timestamp timevar {
    if (timvar>max){setverdict(fail)}
    else {setverdict(pass)}
};
```
Temporal Control

- ... at any place during test case execution,

```c
wait(timepoint);
```

- and similar for message timing

```c
wait(timepoint);
p.send(t);
```
Double check the timing of test system behavior

```
// test system to slow
wait(timepoint);
p.send(MSG_1);
if(now >= timepoint + tol) setverdict(error);

// SUT to slow
wait(timepoint);
p.send(MSG_1);
if(now >= timepoint + tol) setverdict(fail);

// SUT or test system to slow
wait(timepoint);
p.send(MSG_1);
if(now >= timepoint + tol) setverdict(inconclusive);
```
Use Case: Test of an Indicator

Testing Temporal Properties

- Maximum activation time 60 ms, phase length 600 ms
- Synchronization between signals: distance < 5 ms
testcase tcl( ) runs on IndicatorTestComponent{
  var float l_actv, r_actv, f_actv;
  const float TMAX = 0.06;
  activate(tc_timout);
  leverIn.send (LEFT);
  l_actv:= now;
  l_actv:= now;
  interleave{
    [ ] FrontOut.receive(ON) -> timestamp f_actv;
    [ ] RearOut.receive(ON) -> timestamp r_actv;
  }
  if ((f_actv-l_actv > TMAX)
    or (f_actv-r_actv > TMAX)){setverdict(fail)}
  setverdict(pass);
}
testcase tc2( ) runs on IndicatorTestComponent{
    var float r_actv, f_actv;
    const float TMAX = 0.005;
    activate(tc_timeout);
    leverIn.send(LEFT);
    interleave{
        [ ] FrontOut.receive(ON) -> timestamp f_actv;
        [ ] RearOut.receive(ON) -> timestamp r_actv;
    }
    if (abs(r_actv-l_actv) > TMAX){setverdict(fail)}
    setverdict(pass);
}
Summary and Outlook

- RT concepts are tightly integrated with TTCN-3 and
  - provide means for an exact measurement, comparison and verification of the timing of incoming messages, and
  - enables the detection of timing problems during test execution and message dispatching

- Implementation of Concepts
- Integration with high level modeling techniques (i.e. declarative approaches to specify timing constraints).
- Definition of coding and design guidelines to support the RT-capabilities of the newly introduced TTCN-3 concepts.