Testing Real-Time Embedded Systems Using UppAal-TRON - Tool and Application

Kim G. Larsen, Marius Mikucionis, Brian Nielsen, Arne Skou

Aalborg University, DK

{kgl | marius | bnielsen | ask}@cs.aau.dk
Agenda

- Automated Model-based Testing
- Testing Framework
  - Timed Automata
  - Environment Modeling
  - Relativized I/O conformance
- Online Testing Algorithm
- Danfoss EKC
- Other Issues
  - Monitoring and Environment Emulation
  - Coverage Measurement
- Demo
- Conclusions & Future Work
Testing Embedded Software

- **Testing**: Execute actual software (system) with controlled inputs and check responses
- To find errors
- To determine risk of release

- 10-20 errors per 1000 LOC
- 30-50 % of development time and cost
- Software and complexity increases
Automated Model-Based Testing

Does the behavior of the (blackbox) implementation comply to that of the specification?
Online Testing

- Test generated and executed event-by-event (randomly), reactivity
- Long Running, deep testing, imaginative
Real-Time Systems

A system where correctness not only depends on the logical order of events but also on their timing.

Real Time System

Environment

sensors

actuators

Controller

Task

Task

Task

System Model

Environment Model

Input

Output

Modelling & Abstraction

Σ
Our Framework

- **UppAal Timed Automata** Network: Env || IUT

!![Diagram of Testing-UPPAAL model with relevant components labeled.]

- **Correct system behavior**
- **Test Oracle**
- **Monitor**

- Complete and sound algorithm
- Efficient symbolic reachability algorithms
- **UppAal-TRON**: Testing Real-Time Systems Online
- Release 1.3 [http://www.cs.aau.dk/~marius/tron/]
Related Work

- Formal Testing Frameworks
  - [Brinksma, Tretmans]
- Real-Time Implementation Relations
  - [Khoumsi’03, Briones’04]
- Symbolic Reachability analysis of Timed Automata
  - [Dill’89, Larsen’97,…]
- Online state-set computation
  - [Tripakis’02]
- Online Testing
  - [Tretmans’99, Peleska’02, Krichen’04]
Sample Test Runs

INFINITELY MANY SEQUENCES!!!!!!!
Sample Cooling Controller

IUT-model

Env-model

Cr
Env. Modeling

- Realism and Guiding
  - $E_M$: Any action possible at any time
  - $E_1$: Only realistic temperature variations
  - $E_2$: Temperature never increases when cooling
  - $E_L$: No inputs (completely passive)

$E_L \subseteq E_2 \subseteq E_1 \subseteq E_M$
Sample Cooling Controller

IUT

Env-model

$C^r \not\equiv_{I_{loc}} E_M \equiv C^r$
Sample Cooling Controller

\[ C'^r \not\equiv \text{loco}_{E_1} C^r \iff 3d < r \]

d.Med?.d.High?.d.Med?.d.Low?.\varepsilon.On, \varepsilon \leq r
Sample Cooling Controller
Non-Determinism

- **Modeling Action uncertainty**
  - A controller switches a relay when a control variable crosses ‘around’ threshold value

- **Modeling Timing uncertainty**
  - A controller switches a relay between 2 and 10 time units
Implementation relation
Relativized real-time io-conformance

Let $P$ be a set of states
- $\mathbb{TTr}(P)$: the set of timed traces from states in $P$
- $P$ after $\sigma$ = the set of states reachable after timed trace $\sigma$
- $\text{Out}(P)$ = possible outputs and delays in $P$

- $\text{i rt-ioco}_e\ s = \text{def}$
  - $\forall \sigma \in \mathbb{TTr}(e): \text{Out}((e,i) \text{ after } \sigma) \subseteq \text{Out}((e,s) \text{ after } \sigma)$

- $\text{i rt-ioco}_e\ s$ iff $\mathbb{TTr}(i) \cap \mathbb{TTr}(e) \subseteq \mathbb{TTr}(s) \cap \mathbb{TTr}(e)$

Intuition, for all relevant environment behaviors
- never produces illegal output, and
- always produces required output in time

~timed trace inclusion
Randomized Online Algorithm

**Algorithm** `TestGenExec (TestSpec)`\textbf{returns} \{\textbf{pass, fail}\}

\[Z := \{\langle l_0, 0 \rangle\}, \]
\textbf{While} \(Z \neq \emptyset\) and \#iterations\:\leq T \textbf{do}\]
\begin{enumerate}
  \item \textbf{if} \(\text{EnvOutput}(Z) \neq \emptyset\) \hspace{1cm} // Offer an input
      \begin{enumerate}
        \item choose randomly \(a \in \text{EnvOutput}(Z)\)
      \end{enumerate}
    \begin{enumerate}
      \item \textbf{send} \(i\) to SUT
        \begin{enumerate}
          \item \(Z := Z \text{ after } a\)
        \end{enumerate}
    \end{enumerate}
  \end{enumerate}

  \item choose randomly \(\delta \in \text{Delays}(Z)\) \hspace{1cm} // Delay and wait for output
    \begin{enumerate}
      \item \textbf{Wait}(\(\delta\))
      \begin{enumerate}
        \item \textbf{if} \(o\) occurred after \(\delta' \leq \delta\) \textbf{then}
          \begin{enumerate}
            \item \(Z := Z \text{ after } \delta'\)
          \end{enumerate}
        \end{enumerate}
      \end{enumerate}
    \item \textbf{if} \(o \notin \text{ImpOutput}(Z)\) \textbf{then return fail}
    \item \(Z := Z \text{ after } o\)
  \end{enumerate}

  \item \textbf{else} \hspace{1cm} // no output within \(\delta\) time
    \begin{enumerate}
      \item \(Z := Z \text{ after } \delta\)
    \end{enumerate}

  \item \textbf{reset IUT}
    \begin{enumerate}
      \item \(Z := \{\langle l_0, 0 \rangle\}\)
    \end{enumerate}
  \end{enumerate}

\textbf{if} \(Z = \emptyset\) \textbf{then return fail} \textbf{else return pass}

\begin{itemize}
  \item \textbf{Sound}
  \item \textbf{Complete as T} \rightarrow \infty
\end{itemize}
Sound & Complete

- TestGenExec is sound
  - Fail verdict $\Rightarrow \neg (I \text{ ioco}_e S)$
- complete
  - $\neg (I \text{ ioco}_e S) \Rightarrow \text{Prob}(\text{Fail}) \rightarrow 1$ as $T \rightarrow \infty$
- (using only unit delays)
- Assuming
  - IUT can be modeled by an input enabled, deterministic, non-blocking IO-TLOTS with isolated outputs
  - Time unit of IUT is known
  - $T\text{Tr}(\text{IUT})$ and $T\text{Tr}(E)$ are closed under digitization
    - LTS induced by $TA$ with only non-strict guards
  - $T\text{Tr}(S)$ closed under inverse digitization
    - LTS induced by $TA$ with only strict guards
State-set computation

- Compute all potential states the model can occupy after the timed trace $\varepsilon_0, i_0, \varepsilon_1, o_1, \varepsilon_2, i_2, o_2, \ldots$

- Let $Z$ be a set of states
  - $Z$ after $a$: possible states after executing $a$ (and $t^*$)
  - $Z$ after $\varepsilon$: possible states after $t^*$ and $\varepsilon$, totaling a delay of $\varepsilon$

- $o$ is a legal output from SUT iff $O$ in ImpOutput($Z$)
- $a$ is a relevant input in Env iff $I$ in EnvOutput($Z$)

- $\varepsilon$ is a permitted delay iff $Z$ after $\varepsilon \neq \emptyset$
- $\varepsilon$ is a relevant delay iff Delays ($Z$)
State-set Computation

- Compute all potential states the model can occupy after the timed trace $\epsilon_0, i_0, \epsilon_1, o_1, \epsilon_2, i_2, o_2, \ldots$

- Let $Z$ be a set of states
  - $Z$ after $a$: possible states after executing $a$ (and $\tau^*$)
  - $Z$ after $\epsilon$: possible states after $\tau^*$ and $\epsilon_i$, totaling a delay of $\epsilon$

\[ l_0 \xrightarrow{a} l_2 \xrightarrow{\tau} l_4 \]
\[ l_0 \xrightarrow{\tau, x:=0} l_1 \]
\[ l_0 \xrightarrow{x:=0} l_3 \]
\[ l_1 \xrightarrow{x\leq7, a} l_1 \]

\[
\{ \langle l_0, x=3 \rangle \} \text{ after } a = \\
\{ \langle l_2, x=3 \rangle, \langle l_4, x=3 \rangle, \langle l_3, x=0 \rangle \}
\]

\[
\{ \langle l_0, x=0 \rangle \} \text{ after } 4 = \\
\{ \langle l_0, x=4 \rangle, \langle l_1, 0 \leq x \leq 4 \rangle \}
\]

- Represent state sets as sets of symbolic states
- Use symbolic reachability
- (similar to model checkers like UppAal)
Symbolic Reachability

- **Zone** is a conjunction of clock constraints of the form:
  \( \{x_i - x_j < c_{ij}\} \cup \{a_i < x_i\} \cup \{x_j < b_j\} \) where \(\prec \in \{\leq, \leq\}\)

- **Difference bound matrix** - compact representation.

- Symbolic state set \( Z = \{\langle l_1, z_1\rangle, \ldots, \langle l_n, z_n\rangle\} \)

- **Action transition**: \( \langle l, z\rangle \xrightarrow{a} \langle l', (z \land g)_r \land I(l')\rangle \) if \( l \xrightarrow{g,a,r} l' \) is an action transition and \( z \land g \neq \emptyset, (z \land g)_r \land I(l') \neq \emptyset \).

- **Delay transition**: \( \langle l, z\rangle \xrightarrow{\delta} \langle l, z^+\delta \land I(l)\rangle \) iff \( z^+\delta \land I(l) \neq \emptyset \).

\[
z = [(y - x \leq 4) \land (y \geq 5) \land (x \leq 3)]
\]
Real-time Online

• Compute all states reachable after timed trace
• Maintain a set of symbolic states in real time!

Specification
TA-network

[Tripakis’02, Krichen’04]
Danfoss EKC Case
Electronic Cooling Controller

Sensor Input
- air temperature sensor
- defrost temperature sensor
- (door open sensor)

Keypad Input
- 2 buttons (~40 user settable parameters)

Output Relays
- compressor relay
- defrost relay
- alarm relay
- (fan relay)

Display Output
- alarm / error indication
- mode indication
- current calculated temperature

Optional real-time clock or LON network module
Industrial Cooling Plants

01/06/2003
Project Goals

- Can we model significant aspects and time constraints?
- Can we test in real-time?
- Is the tool fast enough?
- How do we control and observe target?
  - Existing product
  - Documentation
    - requirements specification
    - users manuals
    - equipment and software for real test execution
    - Meeting and e-mail with Danfoss Engineers
- Continued collaboration
  - Test of new generation controllers being developed
  - Improved test interface
Basic Refrigeration Control

- **Time**
- **setpoint**
- **+differential**
- **differential**
- **setpoint**
- **lowAlarm Deviation**
- **lowAlarm Limit**
- **highAlarm Limit**
- **highAlarm Deviation**

- **start compressor**
- **stop compressor**
- **normal**
- **min restart time not elapsed**
- **min cooling time not elapsed**
- **alarm delay**

The diagram illustrates the control system for maintaining the setpoint within the differential range. If the temperature falls below the **lowAlarm Limit**, the compressor starts. If it exceeds the **highAlarm Limit**, an alarm is triggered. The system also includes a restart mechanism after a specified time has elapsed.
EKC Adaptation 1

- Read and write parameter “database”
- 47 parameters

EKC Software Layering

- Control Software
- Parameter DB (shared variables)
- Device drivers+kernel
- Hardware+Physical I/O

Test Interface

- AK-Online (PC SW)
  - configuration
  - supervision
  - logging

Win32+OLE+VB

LON→GW→RS232
EKC Adaptation 2

Need better test interface!
- Read-only parameters
- Delay and synchronization

Adaptor
- EKC Adaptation 2
- tcp/ip
- LON+rs232
- win32+OLE+VB Solaris/Linux (C++)

TRON Engine
- compressorOn
- setTemp(20)

Par\#4=20.0
- old copy
- new copy

“continuous” readout
- 2 readouts/s

22.3 0 1

22.1 0 1

par\#4=20.0

setTemp(20)

Need better test interface!
Modeling Principles

- Model significant subset
  - Temperature regulation
  - Alarm monitoring
  - Manual and periodic timer based defrosting
- Modular model
- Compute “calculatedTemperature” in model
  - derive output events from that
  - could be monitored in adaptor
- Environment temperature generators
- Use non-determinism
  - Timing tolerances
  - Model adapter delay and timing uncertainty
Temperature Tracking

"periodic" weighted average: \( T_n = \frac{T_{n-1} \times 4 + T_{\text{sampled}}}{5} \)

- EKC calculated temperature
- Model calculated temperature
- Error/uncertainty envelope

Tolerance in value computation
Tolerance in sampling time
CompressorOn!
Main Model Components

- 18 concurrent timed automata
- 14 clocks, 14 integers
Reverse Engineering

- Unclear and incomplete specifications

Method of Working

1. Formulate hypothesis model
2. Test
3. **FAIL**-verdict ⇒ Refine model
4. **(PASS)** ⇒ Confirm with Danfoss

- Detects differences between actual and modeled behavior

- *Indicates promising error-detection capability*

- 4 examples
Ex1: Control Period

- Control actions issued when "calculatedTemp" crosses thresholds

"periodic" weighted average: \[ T_n = \frac{T_{n-1} \times 4 + T_{\text{sampled}}}{5} \]

- No requirements on period given
- Tested to be 1.2 seconds
Ex2: High Alarm Monitor v1

Clearing the alarm do not switch off alarm state, only alarm relay
Ex2: High Alarm Monitor v2

- Add HighAlarmDisplay action
- Add location for “noSound, but alarmDisplaying”
- (Postpone alarms after defrosting)
Ex3: Defrosting and Alarms

- When defrosting the temperature rises
- Postpone high temperature alarms during defrost
- System parameter alarmDelayAfterDefrost
- Several Interpretations
  1. Postpone alarmDelayAfterDefrost + alarmDelay after defrost?
  2. Postpone alarmDelayAfterDefrost + alarmDelay after highTemp detected?
  3. Postpone alarmDelayAfterDefrost until temperature becomes low; then use alarmDelay

- Option 3 applies!
Ex4: Defrost Time Tolerance

- Defrost relays engaged earlier and disengaged later than expected
- Assumed 2 seconds tolerance
- Defrosting takes long time
- Implementation uses a low resolution timer (10 seconds)
Example Test Run
(log visualization)
State-set Evolution

Correlation between state-sets and model behavior
Cost of state-set update

Average after Delay CPU time, microseconds

Number of Symbolic states

Initial state set size
Testing = Environment emulation + monitoring
Testing

- Replace Systems Real Environment by Tester
- Tester provides inputs
- Tester observes outputs

"Formal Relativized i/o conformance" Relation

- Relevant input event sequences
- Load model

Correct system behavior
- Test Oracle
- Monitor
Environment Emulation

- Compute inputs from environment model
  - Relevant input event sequences
  - Load model
- Feedback or one-way
- Outputs may go to real-system

"Formal Relativized i/o conformance" Relation
Monitoring

- Passively check communication between system and its real environment
  - check system behavior
- Passive Testing

"Formal Relativized i/o conformance" Relation
Measuring Coverage
Coverage Measurements

- How thorough has testing been??
- Idea:
  - Use a model checker, e.g. UppAal
  - Convert timed trace observed during test run to a timed automata (trace automata)
  - Replace Environment by trace automaton
  - Perform Reachability Analysis on annotated model (according to coverage criteria)
Location Coverage

- Test sequence traversing all locations
- Encoding:
  - Enumerate locations $l_0, \ldots, l_n$
  - Add an auxiliary variable $l_i$ for each location
  - Label each ingoing edge to location $i$ \( l_i := \text{true} \)
  - Mark initial visited \( l_0 := \text{true} \)
- Check: \( \text{EF} ( l_0 = \text{true} \land \ldots \land l_n = \text{true} ) \)
Edge Coverage

- Test sequence traversing all edges
- Encoding:
  - Enumerate edges $e_0, \ldots, e_n$
  - Add auxiliary variable $e_i$ for each edge
  - Label each edge $e_i := \text{true}$
- Check: $\text{EF}(e_0=\text{true} \land \ldots \land e_n=\text{true})$
Coverage of non-deterministic models

- Trace 10.a!.5.b?

- Edge $i$ possible covered (is some run)
  - Check: $\text{EF}( e_i=true \land t.end)$

- Edge $i$ definitely covered (in all runs)
  - Check: $\text{AF}(t.end \Rightarrow e_i=true)$

- Edge $i$ definitely not covered (in no runs)
  - Check: $\text{AF}(t.end \Rightarrow e_i=false)$
Demo
Touch-Sensitive Light-Controller

• Patient user: Wait=∞
• Impatient: Wait=15
Touch-sensitive Light-Controller Model
Mutants

• M1 incorrectly implements switch

```java
synchronized public void handleTouch() {
    if(lightState==lightOff) {
        setLevel(oldLevel);
        lightState=lightOn;
    }
    else { //was missing
        if(lightState==lightOn){
            oldLevel=level;
            setLevel(0);
            lightState=lightOff;
        }
    }
}
```

• M2 incorrect additional delay in dimmer as if x:=0 was on ActiveUP ↔ ActiveDN transitions
Conclusions

- Can accurately model EKC-like devices
- Can create models suitable for online testing
- Complete and detailed model not required
  - Select aspects
  - Abstraction
- MBT feasible even if specification is incomplete/unclear
- Promising error-detection capabilities
  - Differences between actual and specified behavior in industrial case
  - Academic mutation studies
- Excellent performance
- Very non-deterministic models causes very large state-sets which can become a computational bottleneck
- Real-time synchronization of IUT and tester is problematic
Future Work

✓ Tool Improvements
  ✓ Import test trace into UppAal
  ✓ Edge & location-coverage measurements
  ✓ Graphical User-Interface
  ✓ Separate environment simulation and monitoring

✗ Further Danfoss Collaboration
  ▪ Better test interface
  ▪ Test newly developed product

✗ Coverage Guiding & RT-criteria

✗ Automatic test adaptation abstraction

✗ Testing Hybrid and Stochastic Systems
Research Challenges

- Modelling
  - How to model real-time systems easily, and quickly, precisely, expressively, ...
  - What is a good formal notion of correctness?

- Tool implementation
  - How to analyze these models?
  - How to compute state-set estimation in real-time?
  - Real-time execution and clock synchronization with IUT?!!

- Robustness
  - Partial observability and uncertainty

- Guiding
  - Can we improve obtained coverage of model??
  - Real-time coverage criteria??
  - Is it efficient in finding errors?

- How to apply in industrial practice?

- Extensions
  - Probabilistic performance testing?
  - Hybrid systems
END