CISS – Center For Embedded Software Systems

Regional ICT Center (2002-)

- 3 research groups
  - Computer Science
  - Control Theory
  - Hardware
  - Wireless Communication

- 20 Employed
- 25 Associated
- 20 PhD Students
- 50 Industrial projects
- 10 Elite-students
- 140+ MDKK

- ARTIST Design
- ARTEMIS
- ... ...

Characteristica:
- Dedicated function
- Complex environment
- SW/HW/Mechanics
- Networked
- Autonomous
- Ressource constrained:
  - Energy
  - Bandwidth
  - Memory
  - ...
- Timing constraints
Model Checking & Performance Analysis
Origin of UPPAAL

TAU
- CCS & Modal Transition Systems
- Refinements
- Modal Mu-Calculus
- Explicit State Representation
- Prolog

EPSILON
- TCCS
- Timed Refinements
- Timed Mu-Calculus
- Regions
- Prolog

UPPAAL
- Timed Automata
- TCTL
- Zones
- C++ & Java

UP4ALL

CAV Award

Contributors

@UPPsala
- Wang Yi
- Paul Pettersson
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun

@AALborg
- Kim G Larsen
- Alexandre David
- Gerd Behrman
- Arne Skou
- Brian Nielsen
- Jacob I. Rasmussen
- Marius Mikucionis
- Thomas Chatain

@Elsewhere
- Emmanuel Fleury, Didier Lime, Johan Bengtsson, Fredrik Larsson, Kåre J Kristoffersen, Tobias Amnell, Thomas Hune, Oliver Möller, Elena Fersman, Carsten Weise, David Griffioen, Ansgar Fehnker, Frits Vandraager, Theo Ruys, Pedro D’Argenio, J-P Katoen, Jan Tretmans, Judi Romijn, Ed Brinksma, Martijn Hendriks, Klaus Havelund, Franck Cassez, Magnus Lindahl, Francois Laroussinie, Patricia Bouyer, Augusto Burgueno, H. Bowmann, D. Latella, M. Massink, G. Faconti, Kristina Lundqvist, Lars Asplund, Justin Pearson...
UPPAAL Model Checker

Editor
- Discrete Control
- Concurrency
- Continuous Aspects
- Stochasticity
- Timing Constraints
- Resources

Simulator
- Performance Analyses

Verifier

Kim Larsen [6]
**Timed Automata**

- **Add a clock** $x$
- Synchronizing action
- Clock Guard Conjunctions of $x \sim n$
- $x$: real-valued clock

Diagram:
- States: Off, Light, Bright
- Transitions:
  - From Off, press? to Light, $x=0$
  - From Light, press? to Bright, $x \leq 3$
  - From Bright, press? to Bright, $x > 3$

Notes:
- [Alur & Dill’89]

Semantics in UPPAAL
Train Crossing

- Safe
- Approaching
- Crossing
- Safe

Tracks:

- River
- Bridge

Time:

- 0
- 20
- 3 – 5

FM Forum -- Model Checking in Action -- Kim G Larsen
Train Crossing

Safe → Approaching → Crossing → Safe
Safe → Approaching → Crossing → Safe

Stop the train while it still stoppable!

FM Forum -- Model Checking in Action -- Kim G Larsen
Train Crossing

Safe → Approaching

Add **timing** + synchronization

Stopped

Safe → Crossing

→ Restarted

Crossing

Safe → Start

Separate diagrams for timing and synchronization.
Editor

GUl

- Unlimited undo and redo
- Syntax and bracket highlighting
- Rectangular selection
- Customization of colors
- Tool tip
- Hiding of information
- Improved help menu with search component

Language
- User defined functions (C-like)
- New types (records, type declarations, meta variables, scalars)
- Partial instantiation of templates
- Select clauses on edges
- Forall and exist quantifiers
Concrete Simulator

Graphical Simulator
- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- Gantt Charts
Symbolic Simulator

Graphical Simulator
- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- Gantt Charts
Verifier

- Exhaustive & automatic checking of requirements
- .. including validating, safety, liveness, bounded liveness and response properties
- .. performance properties, e.g. probabilistic and expectation.
- .. generation of debugging information for visualisation in simulator.
- .. plot composer
Demo
## Evolution of Performance

<table>
<thead>
<tr>
<th>Version</th>
<th>CSMA5</th>
<th>CSMA7</th>
<th>CSMA12</th>
<th>Fischer5</th>
<th>Fischer7</th>
<th>Fischer12</th>
<th>HDDI7</th>
<th>HDDI12</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0.39</td>
<td>8.4 s</td>
<td>—</td>
<td>—</td>
<td>4.2 s</td>
<td>—</td>
<td>—</td>
<td>36.3 s</td>
<td>—</td>
</tr>
<tr>
<td>7.2 MB</td>
<td>—</td>
<td>—</td>
<td>10.6 MB</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>20.1 MB</td>
<td>—</td>
</tr>
<tr>
<td>3.2.12</td>
<td>0.3 s</td>
<td>417 s</td>
<td>—</td>
<td>1.6 s</td>
<td>—</td>
<td>—</td>
<td>7.2 s</td>
<td>—</td>
</tr>
<tr>
<td>3.8 MB</td>
<td>145 MB</td>
<td>—</td>
<td>6.8 MB</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>11.9 MB</td>
<td>—</td>
</tr>
<tr>
<td>3.3.25</td>
<td>0.2 s</td>
<td>198 s</td>
<td>—</td>
<td>1.1 s</td>
<td>—</td>
<td>—</td>
<td>3.2 s</td>
<td>—</td>
</tr>
<tr>
<td>3.4 MB</td>
<td>113 MB</td>
<td>—</td>
<td>6 MB</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>8.4 MB</td>
<td>—</td>
</tr>
<tr>
<td>3.4.6</td>
<td>&lt;0.1 s</td>
<td>40.7 s</td>
<td>—</td>
<td>0.3 s</td>
<td>4706 s</td>
<td>—</td>
<td>0.1 s</td>
<td>5.3 s</td>
</tr>
<tr>
<td>3.1 MB</td>
<td>34.5 MB</td>
<td>—</td>
<td>4.9 MB</td>
<td>267 MB</td>
<td>—</td>
<td>—</td>
<td>1.6 MB</td>
<td>14.1 MB</td>
</tr>
<tr>
<td>4.0.11</td>
<td>&lt;0.1 s</td>
<td>0.2 s</td>
<td>33.8 s</td>
<td>&lt;0.1 s</td>
<td>0.4 s</td>
<td>418 s</td>
<td>&lt;0.1 s</td>
<td>0.4 s</td>
</tr>
<tr>
<td>1.6 MB</td>
<td>38 MB</td>
<td>115 MB</td>
<td>1.6 MB</td>
<td>38.1 MB</td>
<td>300 MB</td>
<td>1.6 MB</td>
<td>38 MB</td>
<td></td>
</tr>
<tr>
<td>4.1.2</td>
<td>&lt;0.1 s</td>
<td>0.2 s</td>
<td>41.9 s</td>
<td>&lt;0.1 s</td>
<td>0.3 s</td>
<td>341 s</td>
<td>0.05 s</td>
<td>0.2 s</td>
</tr>
<tr>
<td>1.6 MB</td>
<td>21.6 MB</td>
<td>99 MB</td>
<td>21 MB</td>
<td>21.6 MB</td>
<td>248 MB</td>
<td>1.6 MB</td>
<td>22.9 MB</td>
<td></td>
</tr>
</tbody>
</table>
Evolution of Code Base

Client-Server Architecture
GUI: Java
Engine: C++
Platforms:
Linux, MacOS, Solaris, Windows

3 major cycles.
THE "secret" of UPPAAL
DBMs [dill89, rokicki93, low95, bengtsson02] are efficient data structures to represent clock constraints in timed automata [ed90]. They are used in UPPAAL [ly97, ly04, bdl04] as the core data structure to represent time. The library features all the common operations such as up (delay, or future), down (past), general updates, different extrapolation functions, etc., on DBMs and federations. The library also supports subtractions. The API is in C and C++. The C++ part uses active clocks and hides (to some extent) memory management.

References

UPPAAL as a back-end

- Formalising the ARTS MPSOC Model in UPPAAL, 2007
- Yggdrasil: Statechart to UPPAAL, 2003
- Component-Based Design and Analysis of Embedded Systems with UPPAAL PORT, 2008
- Verification of COMDES-II Systems Using UPPAAL with Model Transformation, 2008

... ...
Industrial Usage

some examples
Bang & Olufsen (1997)

- Bug known to exist for 10 years
- Ill-described:
  
  2.800 loc +
  3 flowchart +
  1 B&O eng.
- 3 months for modeling.
- UPPAAL detects error with 1.998 transition steps (shortest)
- Error trace was confirmed in B&O laboratory.
- Error corrected and verified in UPPAAL.
- Follow-up project.
Conclusions

- It is possible to make a feasible abstraction of the existing system
- Bugs were found during model development and simulation
- A timing problem was identified during model checking. B&O changed their desing to remove the problem
- Time slicing and interrupt priorities can be modelled by timed automaton
- B&O obtained more confidence in the design before starting their implementation work. The design was robust in the sense that it did not have to be changed during the implementation phase
\[
\begin{align*}
\text{GearControl@Initiate} & \leadsto_{\leq 1500} ( \text{ErrStat} = 0 ) \Rightarrow \text{GearControl@GearChanged} \\
\text{GearControl@Initiate} & \leadsto_{\leq 1000} \\
& \quad ( \text{ErrStat} = 0 \land \text{UseCase} = 0 ) \Rightarrow \text{GearControl@GearChanged} \\
\text{Clutch@ErrorClose} & \leadsto_{\leq 200} \text{GearControl@CCloseError} \\
\text{Clutch@ErrorOpen} & \leadsto_{\leq 200} \text{GearControl@COpenError} \\
\text{GearBox@ErrorIdle} & \leadsto_{\leq 250} \text{GearControl@GSetError} \\
\text{GearBox@ErrorNeu} & \leadsto_{\leq 200} \text{GearControl@GNeuError} \\
\text{Inv} ( \text{GearControl@CCloseError} \Rightarrow \text{Clutch@ErrorClose} ) & \\
\text{Inv} ( \text{GearControl@COpenError} \Rightarrow \text{Clutch@ErrorOpen} ) & \\
\text{Inv} ( \text{GearControl@GSetError} \Rightarrow \text{GearBox@ErrorIdle} ) & 
\end{align*}
\]
Advanced Noise Reduction Techniques

Frequency Diversity

FM Forum -- Model Checking in Action -- Kim G Larsen
RESULTS

• Verification that existing round-robin scheduler with the given size of buffers is safe (no underflow, no overflow)
• Synthesis of scheduler for drastically minimized buffer sizes.
• Savings not important enough for methodology to be taken up by Terma!

Adder 1
S = A + S' - A'

Adder 2
T = B + T' - B'

Buffer 1
1 Kbytes

Buffer 2
1 Kbytes

Buffer 3
512 bytes

Buffer 4
2 Kbytes

Buffer 5
512 bytes

Buffer 6
2 Kbytes

Buffer 7
2 Kbytes

Buffer 8
2 Kbytes

Buffer 9
2 Kbytes

Input A
8 (100MHz)

Input B
8 (100 MHz)

Output S

Output T

128 (200 MHz)

SDRAM
Attitude and Orbit Control Software
TERMA A/S Steen Ulrik Palm, Jan Storbank Pedersen, Poul Hougaard
- **Application software (ASW)**
  - built and tested by Terma:
  - does attitude and orbit control, tele-commanding, fault detection isolation and recovery.

- **Basic software (BSW)**
  - low level communication and scheduling periodic events.

- **Real-time operating system (RTEMS)**
  - Priority Ceiling for ASW,
  - Priority Inheritance for BSW

- **Hardware**
  - single processor, a few communication buses, sensors and actuators

---

**Requirements:**
Software tasks should be schedulable.
CPU utilization should not exceed 50% load
<table>
<thead>
<tr>
<th>ID</th>
<th>Task</th>
<th>Specification</th>
<th>Blocking times</th>
<th>WCRT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Period</td>
<td>WCET</td>
<td>Deadline</td>
</tr>
<tr>
<td>1</td>
<td>RTEMS_RTC</td>
<td>10.000</td>
<td>0.013</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>AswSync_SyncPulseIsr</td>
<td>250.000</td>
<td>0.070</td>
<td>1.000</td>
</tr>
<tr>
<td>3</td>
<td>Hk_SamplerIsr</td>
<td>125.000</td>
<td>0.070</td>
<td>1.000</td>
</tr>
<tr>
<td>4</td>
<td>SwCyc_CycStartIsr</td>
<td>250.000</td>
<td>0.200</td>
<td>1.000</td>
</tr>
<tr>
<td>5</td>
<td>SwCyc_CycEndIsr</td>
<td>250.000</td>
<td>0.100</td>
<td>1.000</td>
</tr>
<tr>
<td>6</td>
<td>Rt1553_Isr</td>
<td>15.625</td>
<td>0.070</td>
<td>1.000</td>
</tr>
<tr>
<td>7</td>
<td>Bc1553_Isr</td>
<td>20.000</td>
<td>0.070</td>
<td>1.000</td>
</tr>
<tr>
<td>8</td>
<td>Spw_Isr</td>
<td>39.000</td>
<td>0.070</td>
<td>2.000</td>
</tr>
<tr>
<td>9</td>
<td>Obdh_Isr</td>
<td>250.000</td>
<td>0.070</td>
<td>2.000</td>
</tr>
<tr>
<td>10</td>
<td>RtSdb_P.1</td>
<td>15.625</td>
<td>0.150</td>
<td>15.625</td>
</tr>
<tr>
<td>11</td>
<td>RtSdb_P.2</td>
<td>125.000</td>
<td>0.400</td>
<td>15.625</td>
</tr>
<tr>
<td>12</td>
<td>RtSdb_P.3</td>
<td>250.000</td>
<td>0.170</td>
<td>15.625</td>
</tr>
<tr>
<td>13</td>
<td>FdirEvents</td>
<td>250.000</td>
<td>5.000</td>
<td>230.220</td>
</tr>
<tr>
<td>14</td>
<td>NominalEvents.1</td>
<td>250.000</td>
<td>0.720</td>
<td>230.220</td>
</tr>
<tr>
<td>15</td>
<td>MainCycle</td>
<td>250.000</td>
<td>0.400</td>
<td>230.220</td>
</tr>
<tr>
<td>16</td>
<td>HkSampler_P.2</td>
<td>125.000</td>
<td>0.500</td>
<td>62.500</td>
</tr>
<tr>
<td>17</td>
<td>HkSampler_P.1</td>
<td>250.000</td>
<td>6.000</td>
<td>62.500</td>
</tr>
<tr>
<td>18</td>
<td>Acb_P</td>
<td>250.000</td>
<td>3.000</td>
<td>50.000</td>
</tr>
<tr>
<td>19</td>
<td>IoCyc_P</td>
<td>250.000</td>
<td>3.000</td>
<td>50.000</td>
</tr>
<tr>
<td>20</td>
<td>PrimaryF</td>
<td>250.000</td>
<td>34.050</td>
<td>59.600</td>
</tr>
<tr>
<td>21</td>
<td>RCSControlF</td>
<td>250.000</td>
<td>4.070</td>
<td>239.600</td>
</tr>
<tr>
<td>22</td>
<td>Obt_P</td>
<td>1000.000</td>
<td>1.100</td>
<td>100.000</td>
</tr>
<tr>
<td>23</td>
<td>Hk_P</td>
<td>250.000</td>
<td>2.750</td>
<td>250.000</td>
</tr>
<tr>
<td>24</td>
<td>StsMon_P</td>
<td>250.000</td>
<td>3.300</td>
<td>125.000</td>
</tr>
<tr>
<td>25</td>
<td>TmGen_P</td>
<td>250.000</td>
<td>4.860</td>
<td>250.000</td>
</tr>
<tr>
<td>26</td>
<td>Sgm_P</td>
<td>250.000</td>
<td>4.020</td>
<td>250.000</td>
</tr>
<tr>
<td>27</td>
<td>TcRouter_P</td>
<td>250.000</td>
<td>0.500</td>
<td>250.000</td>
</tr>
<tr>
<td>28</td>
<td>Cmd_P</td>
<td>250.000</td>
<td>14.000</td>
<td>250.000</td>
</tr>
<tr>
<td>29</td>
<td>NominalEvents.2</td>
<td>250.000</td>
<td>1.780</td>
<td>230.220</td>
</tr>
<tr>
<td>30</td>
<td>SecondaryF.1</td>
<td>250.000</td>
<td>20.960</td>
<td>189.600</td>
</tr>
<tr>
<td>31</td>
<td>SecondaryF.2</td>
<td>250.000</td>
<td>39.690</td>
<td>230.220</td>
</tr>
<tr>
<td>32</td>
<td>Bkgnd_P</td>
<td>250.000</td>
<td>0.200</td>
<td>250.000</td>
</tr>
</tbody>
</table>
CONCLUSION

- Schedulability framework made available in UPPAAL
- Provides more exact analysis than classical methods
  - Depending on WCET information the task set is schedulable or not.
- Performance:
  - 1-2 minutes: $\text{BCET} = \text{WCET}$ or $\text{BCET}/\text{WCET} < 0.5$
  - 1 day: $0.5 < \text{BCET}/\text{WCET} < 0.8$
- Work on domain specific notation in order to be fully taken up by company.
Within the Prisma project of TNO-ESI and Philips Lighting, research is done into the robustness and reliability of large-scale indoor lighting systems. The focus is on the robustness of the lighting control system. To analyse control system robustness, model checking is used. Timed automata models of lighting control systems have been created and checked with the model checker Uppaal. To validate the model checking results, the models have been implemented as a prototype lighting control system.
UPPAAL Outside Europe

Performance Analyses

Verifier

FM Forum -- Model Checking in Action -- Kim G Larsen

Kim Larsen [ 37]
Online vs. Offline

Online testing:

**Pros:**
- Abstract system-level behavior
- Realistic setup, many components
- Adaptive, explores only relevant states
- Allows concurrency, non-determinism
- Long and intricate interactions
- Automatic check against model

**Cons:**
- Does not guarantee coverage
- Interpreting model can be slow
- Can be difficult to replicate
- Does not replace offline testing

Offline testing:

- Real-time systems are inherently non-deterministic
- Non-determinism yields exponentially large tests
- Few or no concurrent components
- Short and specific interactions
- Evaluation requires careful assertion programming
Model Interpretation

Test execution
- Inputs translated as test stimuli
- Test responses recognized as outputs

Test generation
How the tester should behave:
- Anything is possible (stress testing)
- Emulate physical processes
- Specific use-case scenario
- Replay a previous test trace (regression)

Test evaluation/monitoring
How the IUT should behave:
- Consume any input at anytime
- Produce outputs expected by model
- Neither too late nor too early
- Non-deterministic:
  - multiple outcomes
  - imprecision of timing
  - concurrency
Yggdrasil (offline)
MBAT Daimler Case (2014)
Test Code & Output

```cpp
DrumScaleOneWeighing_IsInTopPos = 2;

DrumScaleOneWeighing_StartSTM( &me->itsDrumScaleOneWeighing );

HouseSettings_SetRamChar( &me->itsHouseSettings, MS_CURRENT_ACTIVE_STILO, (UCHAR)0 );

me->itsDrumScaleWeighingStable.IsReady_Return = 1;
me->itsDrumScaleRollDrum.IsReady_Return = 1;

Test_Verifies( &me->itsTest, "DrumScaleOneWeighing_running", (UCHAR)IS_IN( "DrumScaleOneWeighing, DrumScaleOneWeighing_running" ) );

Test_Verifies( &me->itsTest, "DrumScaleOneWeighing_StartOneWeighingState", (UCHAR)IS_IN( "DrumScaleOneWeighing, DrumScaleOneWeighing_StartOneWeighingState" ) );

PrintCurrentState( me );

DrumScaleOneWeighing_StartOneWeighing( &me->itsDrumScaleOneWeighing, (FLOAT)12.0, (void (*)(void *))void * const, (LCDW32)OneWeighingFinishedCb, (UCHAR (*)(void *))void * const, OkToRollDrumCb, me );

Test_Verifies( &me->itsTest, "DrumScaleOneWeighing_WeighingDrumEmpty", (UCHAR)IS_IN( "DrumScaleOneWeighing, DrumScaleOneWeighing_WeighingDrumEmpty" ) );

PrintCurrentState( me );

Test_Comment( &me->itsTest, "DrumScaleRollDrum_IsInTopPos is 1", (UCHAR)DrumScaleRollDrum_IsInTopPos( &me->itsDrumScaleRollDrum ) );

me->itsDrumScaleWeighingStable.WeighingFinishedCb( me->itsDrumScaleWeighingStable.owner, 0.6F, 9.62F );
```
Yggdrasil Industrial Use

- Novo Nordisk
  - Reduction in time for testing a module 30 days
    - 30 days → 2 days
- Skov A/S
- TK Validate
  - Ambitious business plan

- Evaluation at
  - Daimler
  - Infinion Austria
  - EADS
  - Bombardier
    - Cov. Inc 40%
    - Reduced test time 20% (80% for unit test)
TRON (online)

Uppaal TRON

TCP/IP socket streams

Java method calls

setLevel(L)

handleGrasp()

handleRelease()
Outcome

4 instances of discrepancy between model and actual behavior, also involving timing errors.

Danfoss has adopted MBT in development of new more complex controller!

- Sequanto SeqZap test harness
- Programmable Logic Controllers (PLC)
Advantages of MBT

- Engineer focus on what to test at a high level of abstraction
- Avoids cost of making scripts
  - As much test code as production code
  - Maintenance nightmare
- Heard of, but is still considered an advanced technique by industry
- Industry is very motivated, MB A&T will give
  - 10% cost reduction
  - 20% quality improvement
**Model Checking & Testing**

**Model Checking**
- Abstract models
- Exhaustive “proof”
- Many mature tools
- Early detection of errors
- State space expl

**Testing**
- Checks the actual implementation
- Only few executions checked
- But is the most direct method

How to effectively combine the different model checking and testing techniques?
UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).

The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University in Denmark.

Download

News: The current official release is UPPAAL 4.0.13 (Sep 27, 2016). Compared to version 3, the 4.0 release is the result of over 2.5 years of additional development, and many new features and improvements are introduced (see also the release note and the web help section for features). To support models created in previous versions of UPPAAL, version 4.0 can convert most old models directly from the GUI (alternatively it can be run in 3.4 compatibility mode by defining the environment variable `UPPAAL_34_COMPAT`, see also Item 2 of the FAQ).

Since Feb 20 2016, we also distribute a development snapshot of the forthcoming UPPAAL 4.2. The current development snapshot version is 4.2.4 released Jul 11, 2016.

License

The UPPAAL tool is free for non-commercial applications in academia. For commercial applications a commercial license is required. Please see the Download section for information.

DESIGN VERIFICATION FOR EMBEDDED SYSTEMS

Our world-leading and internationally acclaimed model-checking tool UPPAAL is now available for commercial use!