Demo : Applications of the CPAL Language to Model, Simulate and Program Cyber-Physical Systems

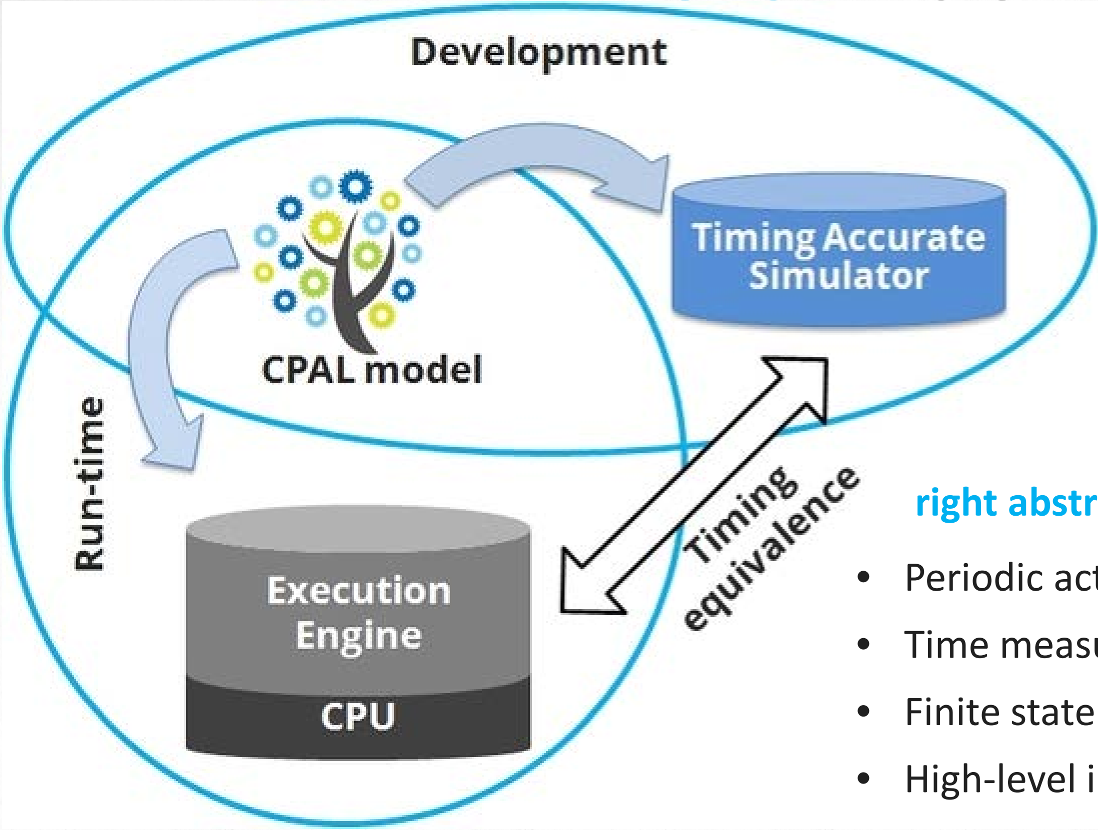
Loïc FEJOZ, Real-time-at-Work

Nicolas NAVET, **Sakthivel M. SUNDHARAM** and Sebastian ALTMEYER, University of Luxembourg

(RTAS 2016)

Vienna, Austria, April 12, 2016

Vision behind CPAL



In CPAL current release, execution order of processes remains the same in simulation and in real-time mode

right abstractions for embedded systems :

- Periodic activities and real-time scheduling
- Time measurements and manipulation
- Finite state machines
- High-level interfaces to I/Os
- Conceived to facilitate the writing of **correct embedded code** (incl. restrictions)

"Write once, Run Anywhere" of Java does not **guarantee** anything about **timing behaviour** on different platforms

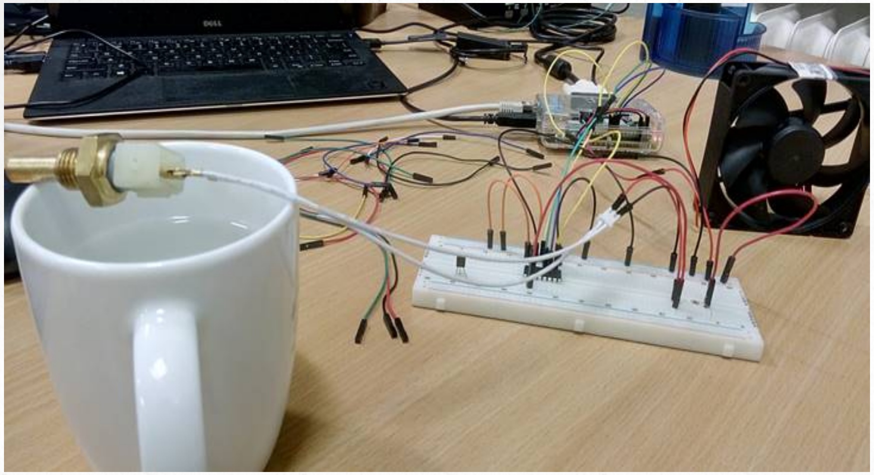
Development environments are unnecessary complex and often expensive and **Model interpretation** brings benefits

Simulating execution times

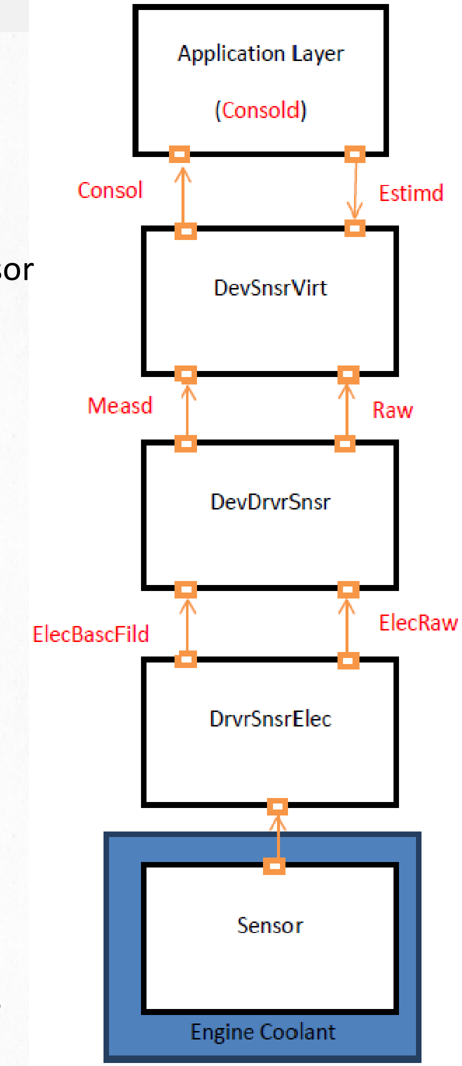
```
processdef OneShortOneLong()
{

  state State1 {
    @cpal:time {
      State1.execution_time = 20ms;
    }
  }
  on (true) to State2;
  state State2 {
    @cpal:time {
      State2.execution_time = 40ms;
    }
  }
  on (true) to State1;
}

process OneShortOneLong: aTask[60ms]();
```

Timing annotations can be derived by built-in monitoring facilities and are respected by the simulator

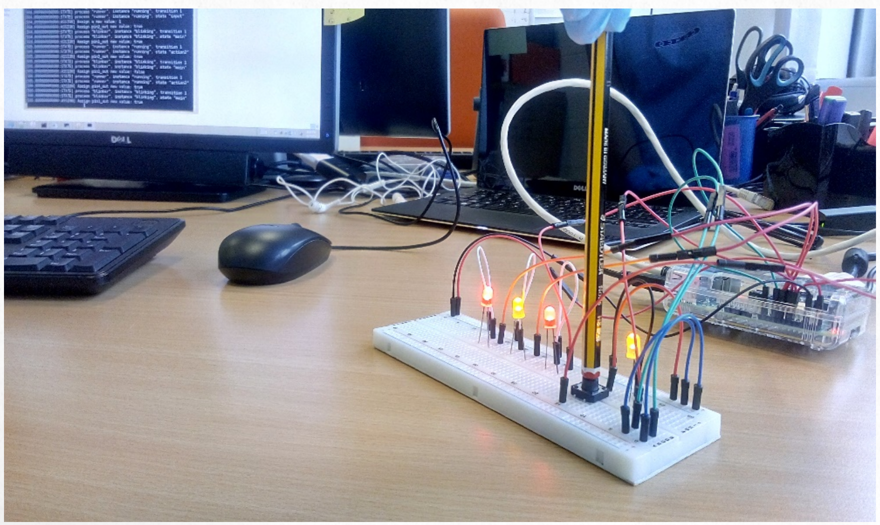
Demo # AUTOSAR Pattern – Engine function



- Raspberry Pi 2 Model B
- CPAL Raspberry Pi [Interpreter]
- Engine coolant temperature Sensor [FAE] and product [info]
- MCP 3008 – External ADC
- SPI Communication between MCP3008 and RASPI
- Script for reading MCP 3008 and pipe-out to CPAL
- CPAL model for engine coolant temperature calculation →



Demo # Event order determinism – Simulation / Real-time



- CPAL has 2 execution modes
- In simulation - Code executed in zero time – except if stated with timing annotations. Interpreter is hosted by OS
- In real-time - Code (instructions, read/write I/Os) takes time to execute –depends on the platform Interpreted, executed by bare hardware or hosted by OS

on Freescale FRDM-K64F:

- ✓ max. activation jitter: 40us
- ✓ timer interrupt: 0.6us
- ✓ context switch overhead: 2us