Une école de l'IMT

Modeling Heterogeneous Embedded Systems with TTool

Daniela Genius, Ludovic Apvrille, Letitia W. Li, Marie-Minerve Louërat, François Pêcheux, Haralampos Stratigopoulos

Daniela.Genius@lip6.fr

IDM, GDR GPL

Design Methodology of an Embedded System

System Partitioning between HW and SW

- Functions → execution nodes
- Communications between functions → communication and storage nodes
- Commonly "system-level partitioning" with (very) abstract models

Design Methodology of an Embedded System

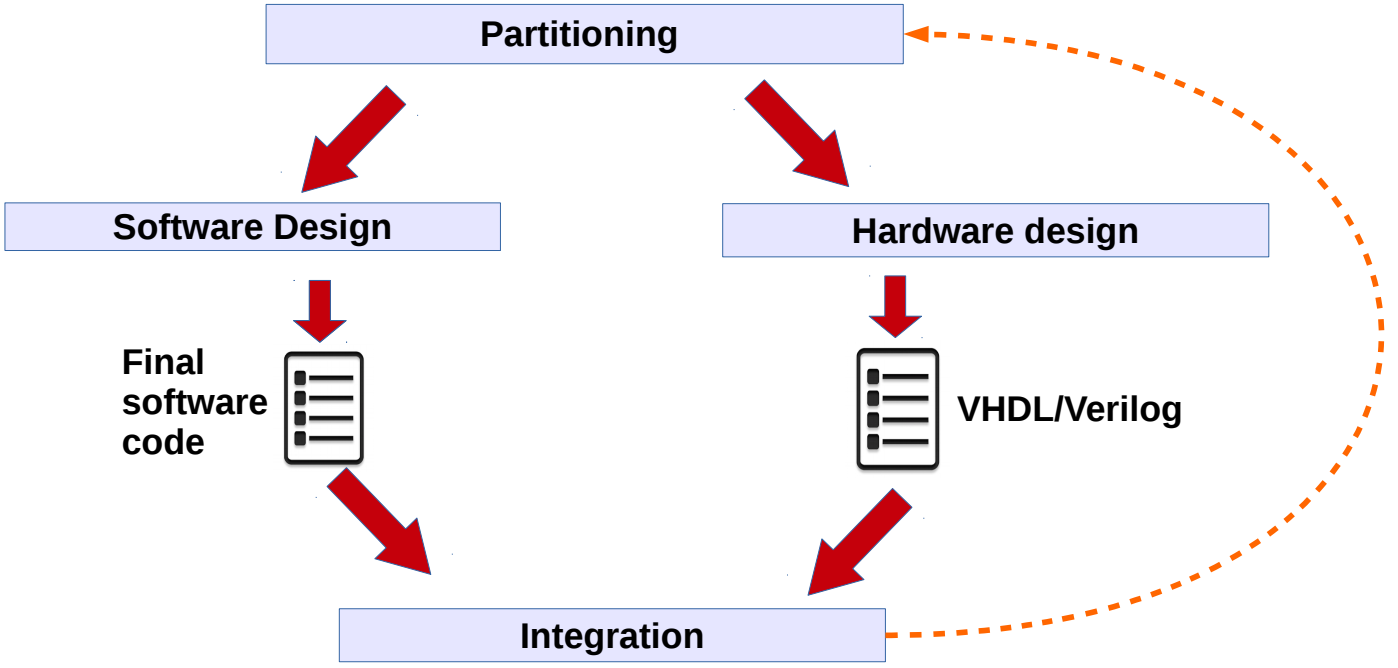
System Partitioning between HW and SW

- Functions → execution nodes
- Communications between functions → communication and storage nodes
- Commonly "system-level partitioning" with (very) abstract models

Software and hardware design

- Software and hardware are designed independently
- Integration phase

Problem: Late Discovery of Bad Partitioning



- Usually focused on *one* modeling aspect: difficult to iterate between partitioning and design
- Objective: model-based approach with close interaction between partitioning and software design

Related Work

- Ptolemy II (Ptolemy.org 2014)
- METROPOLIS (Balarin et al. 2003)
- Sesame and Daedalus (Thompson et al. 2007)
- MARTE with generation for Simics (Taha et al. 2010)
- MDGen (SODIUS)
- B method, Event-B (Abrial 2010)
- AADL (Feiler et al. 2012)

Overall Method

Partitioning with Design Space Exploration techniques (DIPLODOCUS)

Functional view

Architecture view

Mapping view

Reconsideration of partitioning decisions

Software Design and Prototyping (AVATAR)

Software Component

Deployment view

Refinements

Abstractions

Abstractions

Hardware model

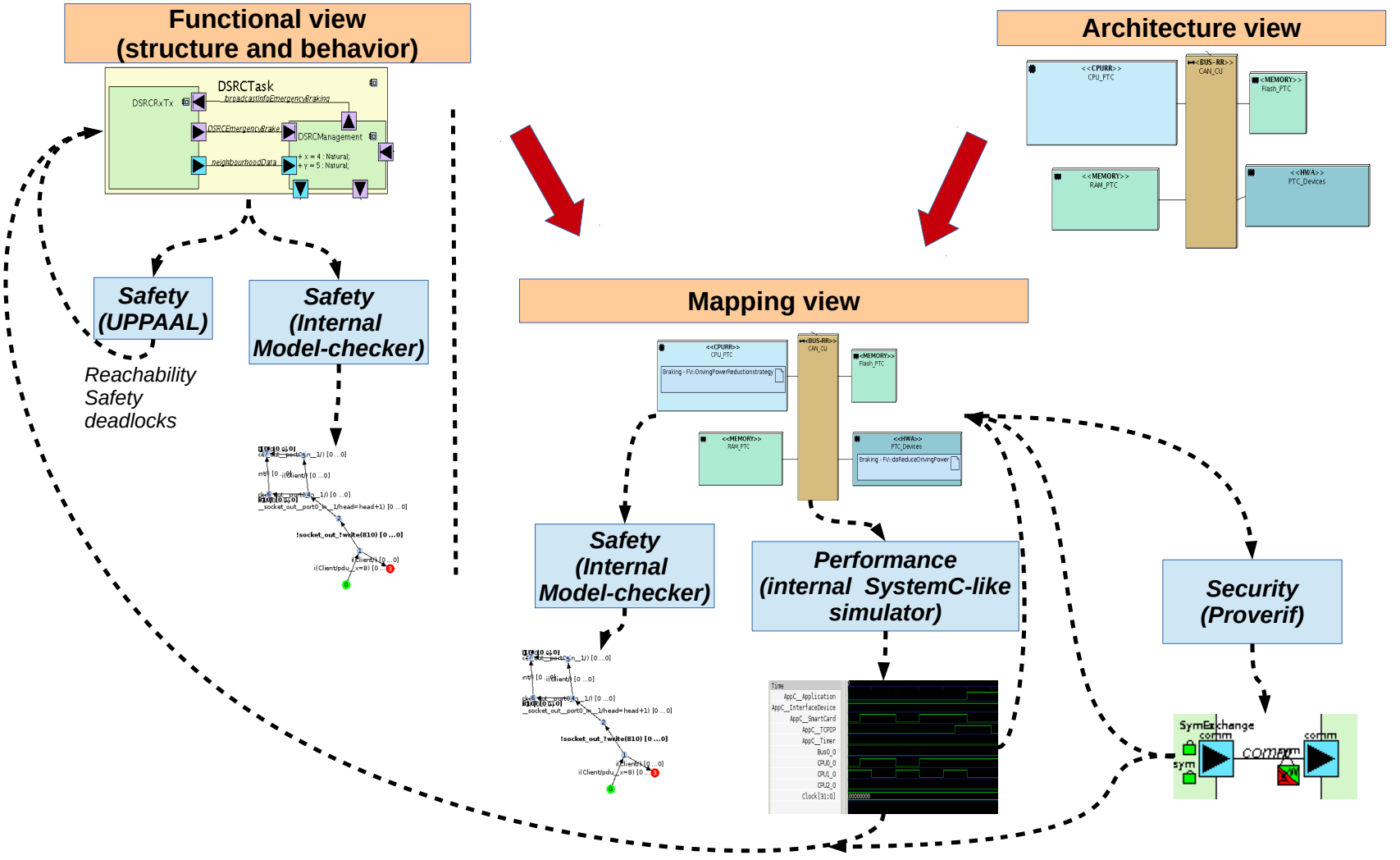
Hardware design

Simulation and Verification : safety, security and performance

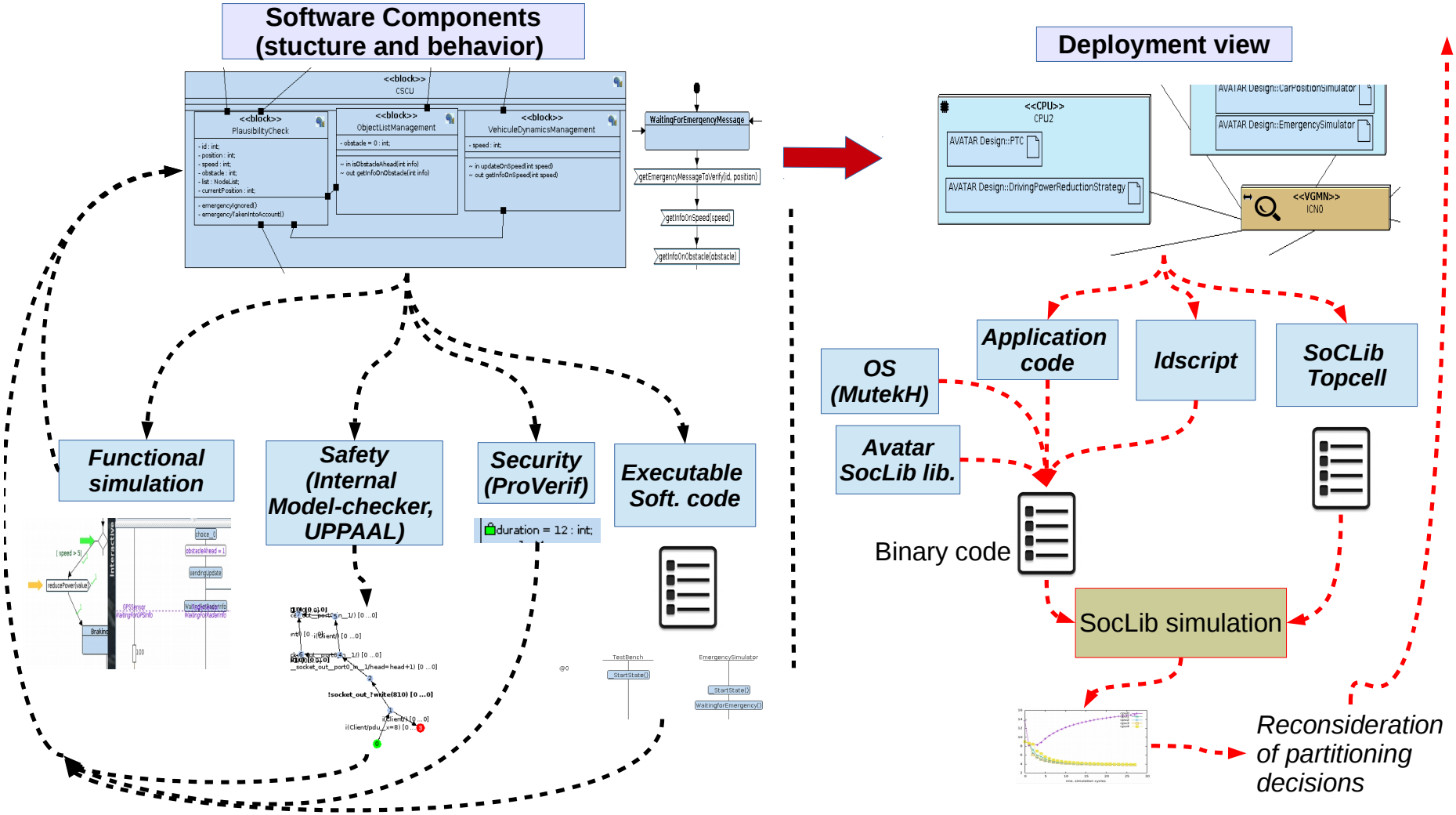
Final software code

VHDL/Verilog

Partitioning Method



Software Design Method

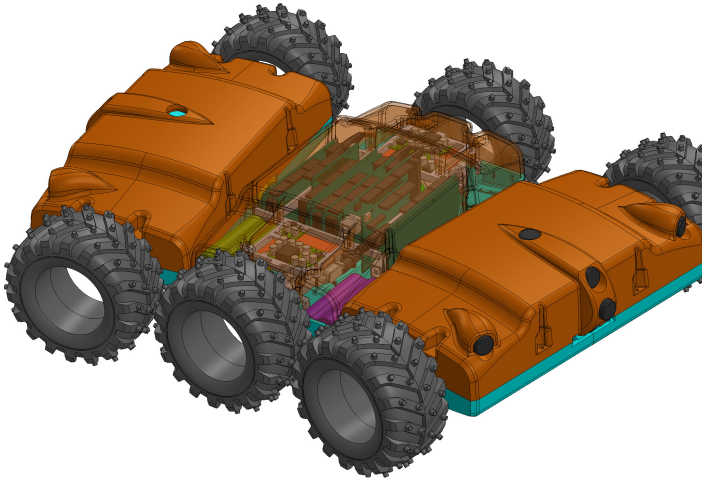


Virtual Prototyping with SoCLib

- Open platform for virtual prototyping of multi-processors System on Chip (MP-SoC)
- Started in 2006 as an ANR project with 11 research labs and 6 industrial partners: LIP6, Bull, CEA-LETI, INRIA, STMicro, ...
- Public domain library of SystemC models of hardware components
- Operating systems: NetBSD, MutekH, GIET, ALMOS, ...
- Different levels of modeling and simulation
  - Transaction Level (TLM)
  - Transaction Level with Time (TLM-T)
  - Cycle Accurate Bit Accurate (CABA)

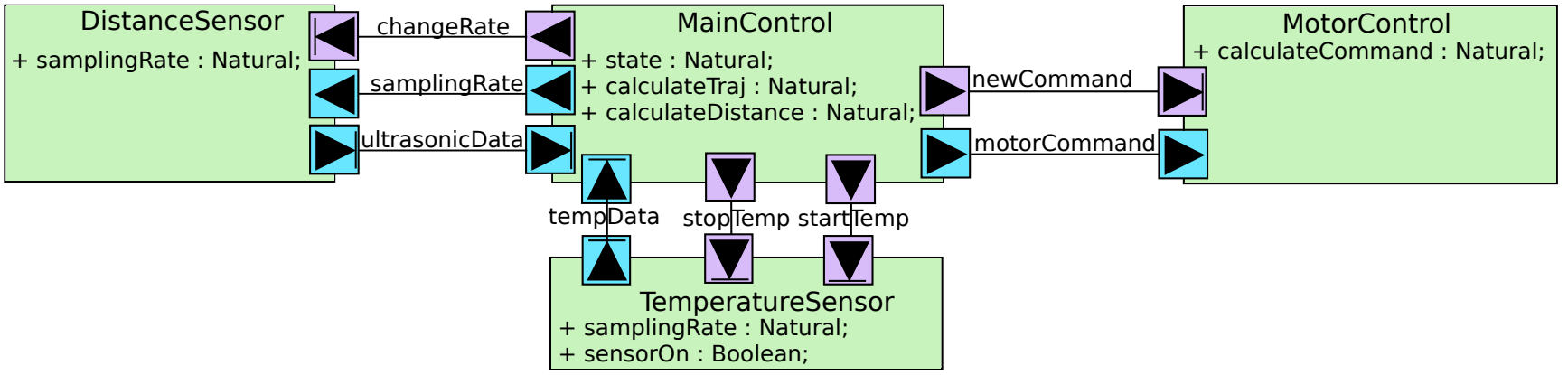
Case Study: Rover

Autonomous vehicle for disaster relief efforts (earthquake)

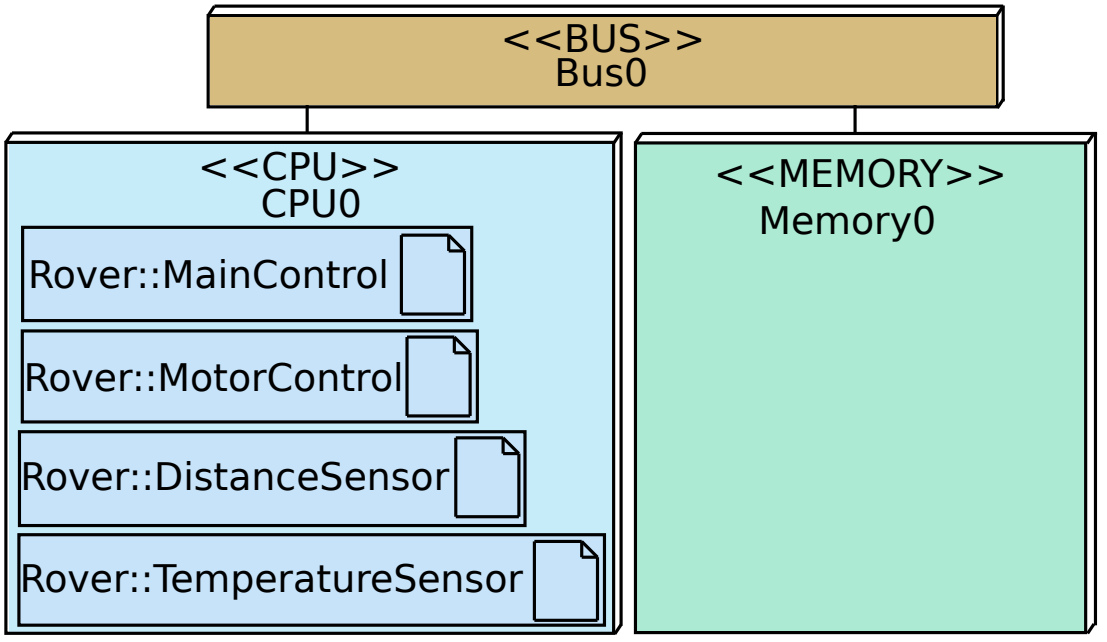


- Telemetric sensors to detect obstacles and navigate terrain autonomously
  - No obstacles in proximity $\rightarrow$ decrease sampling rate
  - Obstacle detected in close proximity $\rightarrow$ increase sampling rate
- Temperature and pressure sensors
- Avoid collisions $\rightarrow$ set time frame $\rightarrow$ impose maximal latency

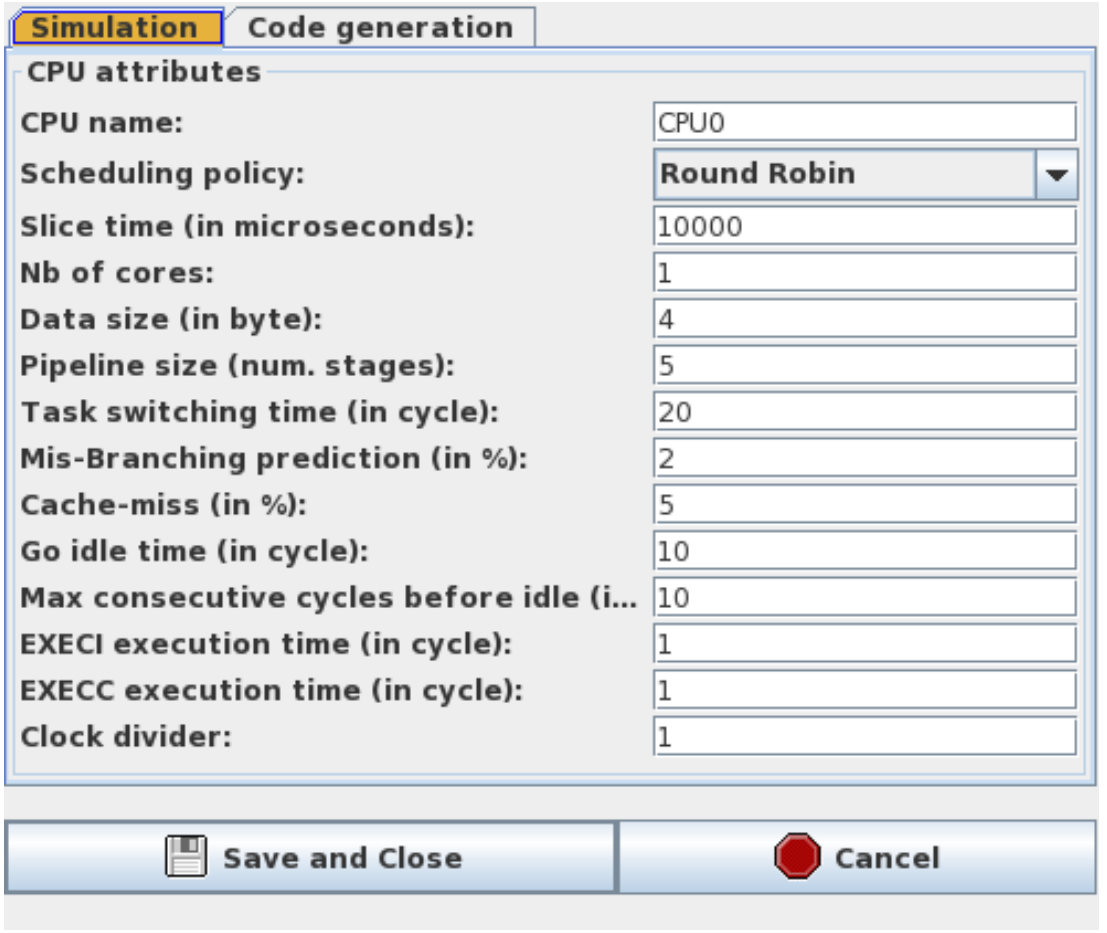
Functional View



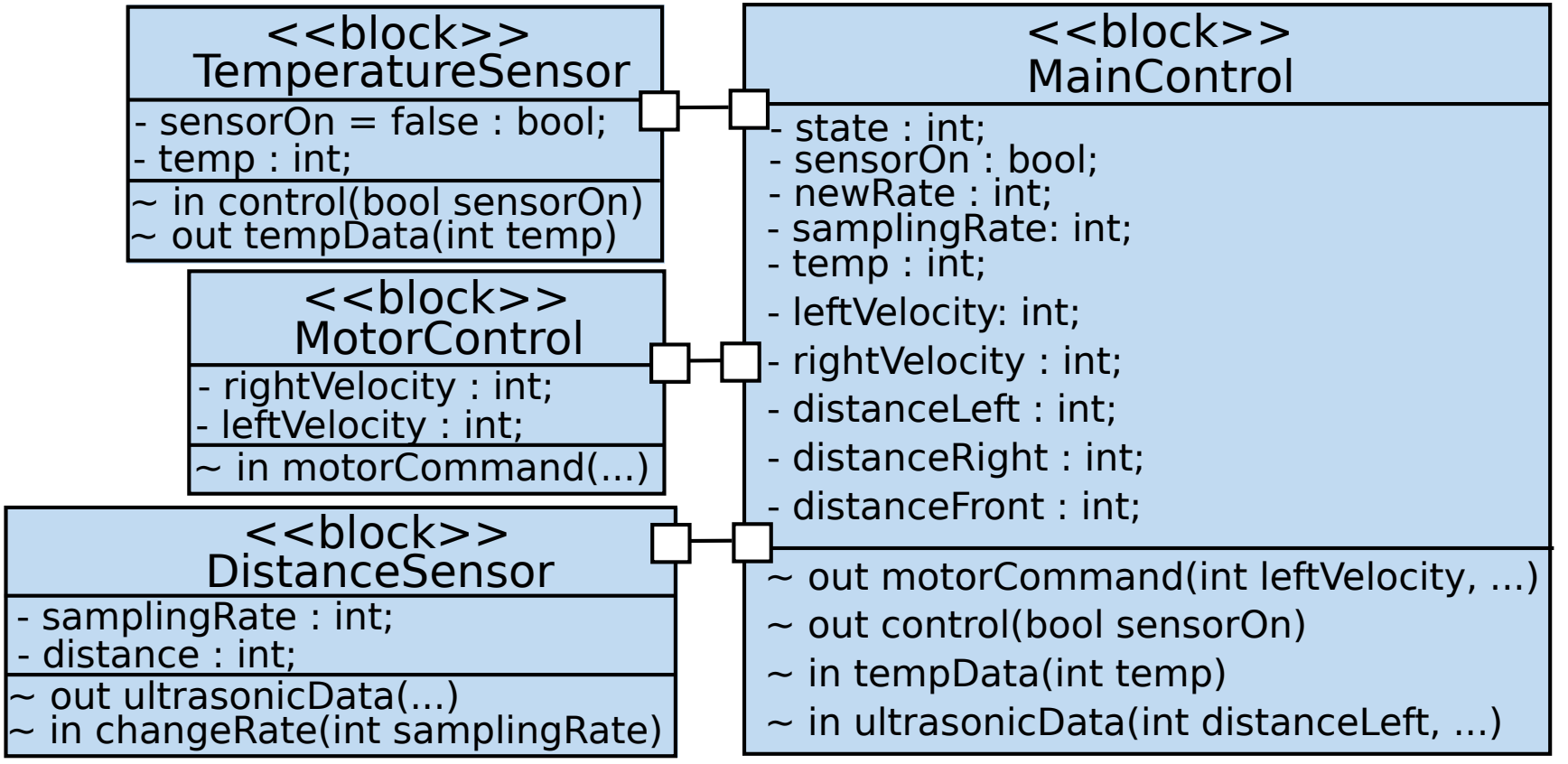
Mapping View



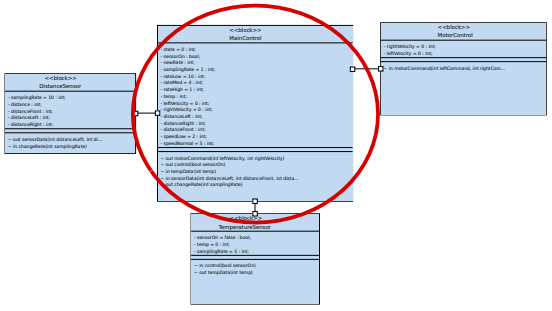
CPU Configuration

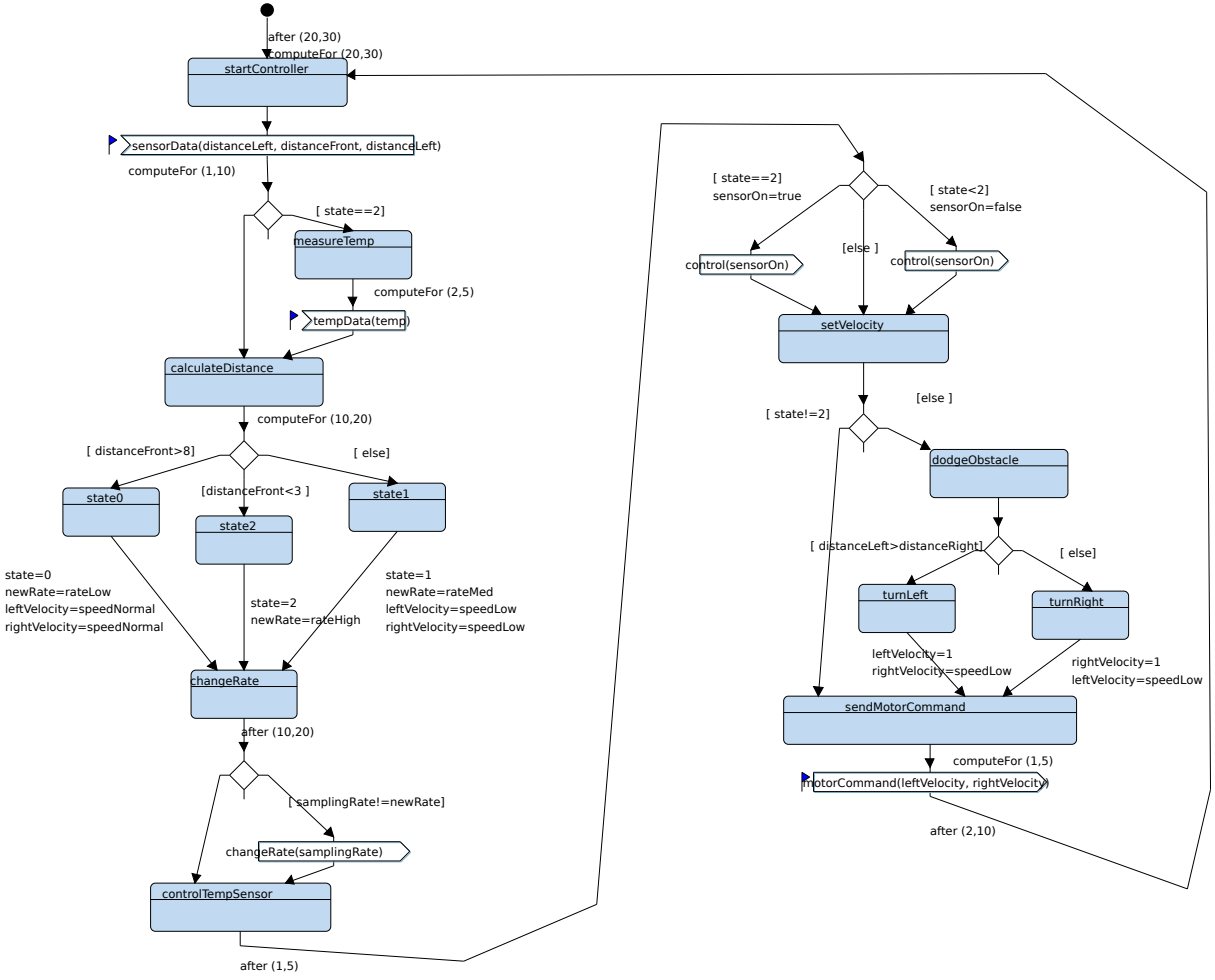


Software Components: Block Diagram

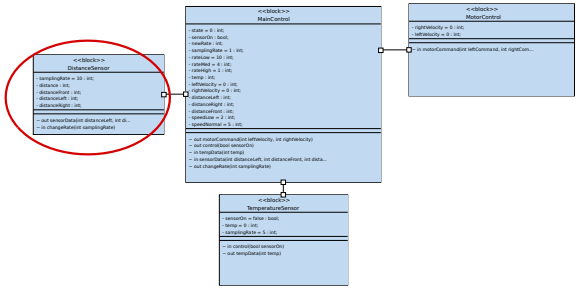


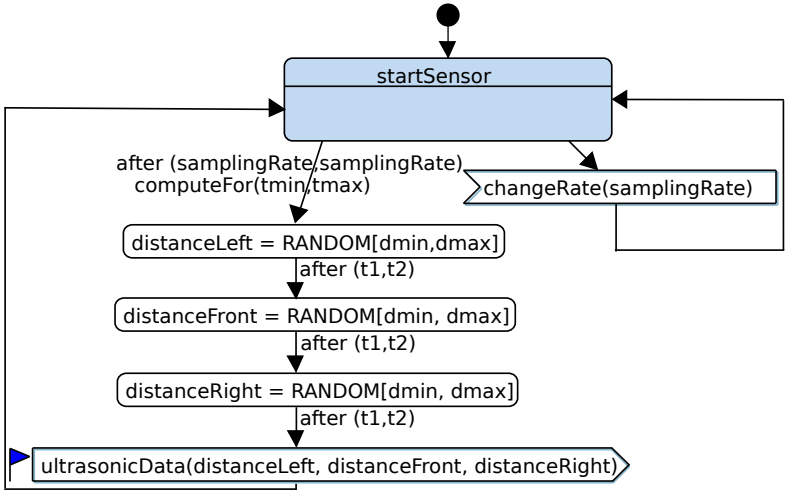
State Machine Diagrams



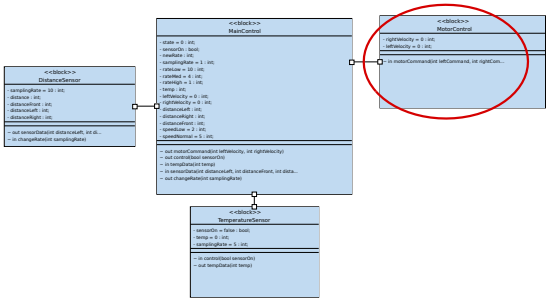


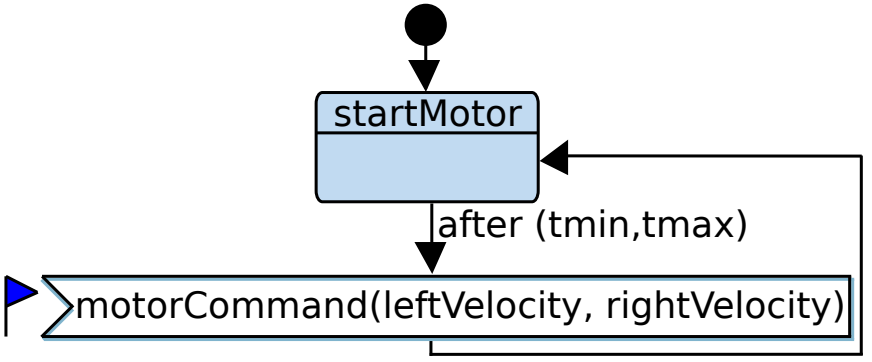
State Machine Diagrams



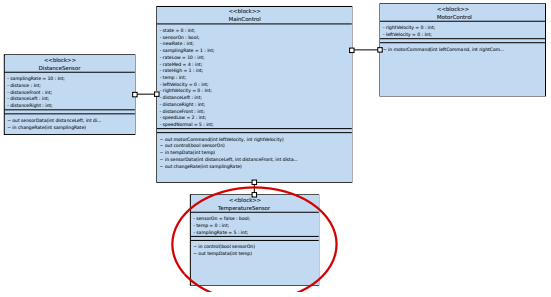


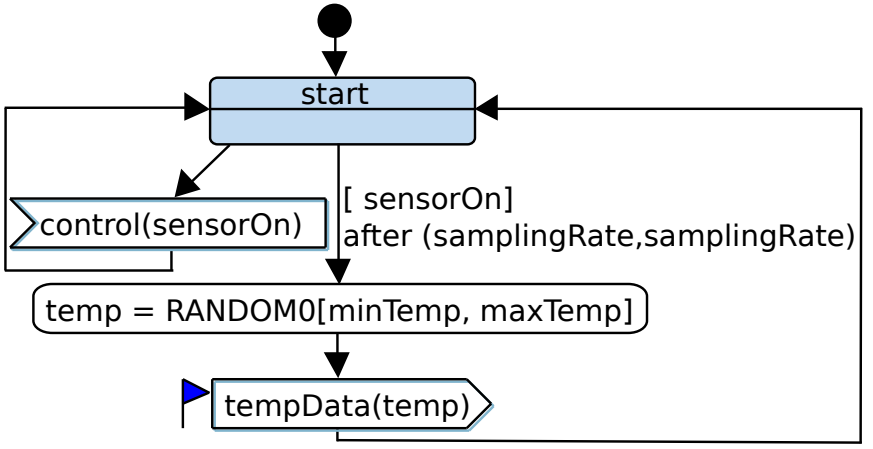
State Machine Diagrams



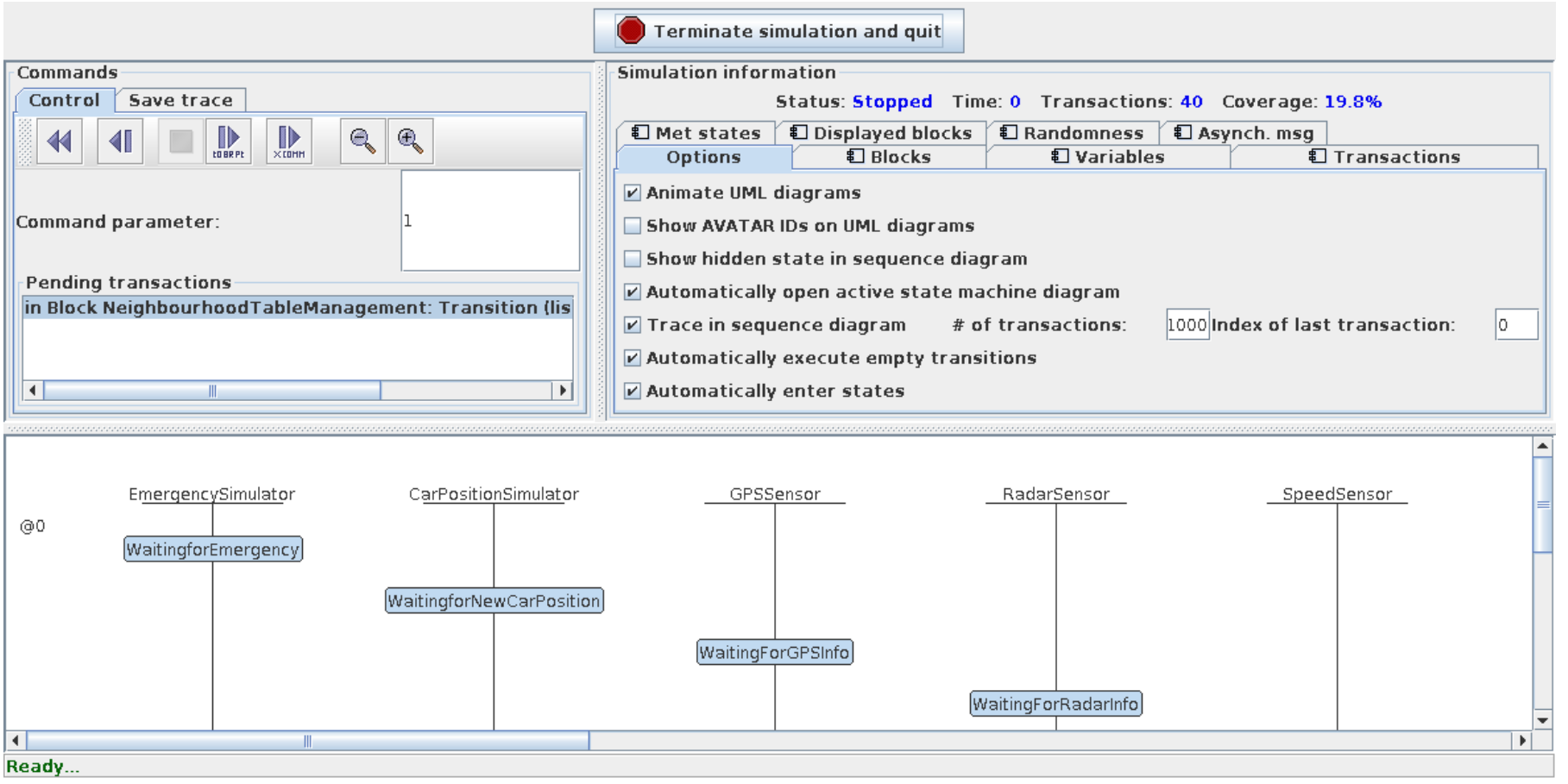


State Machine Diagrams

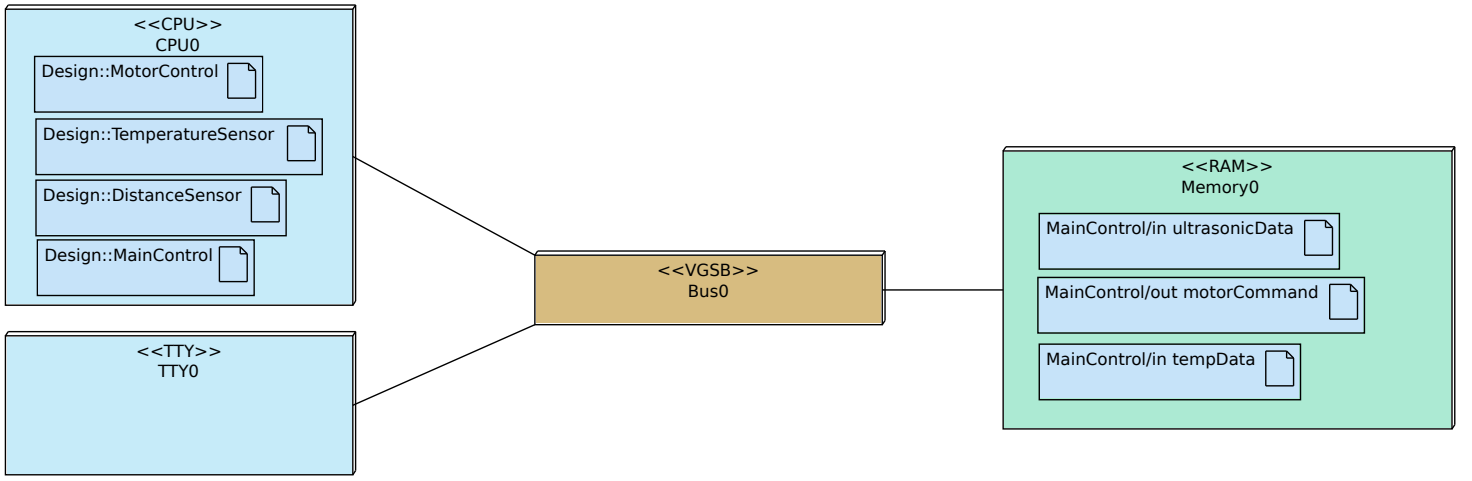




Software Components: Simulation



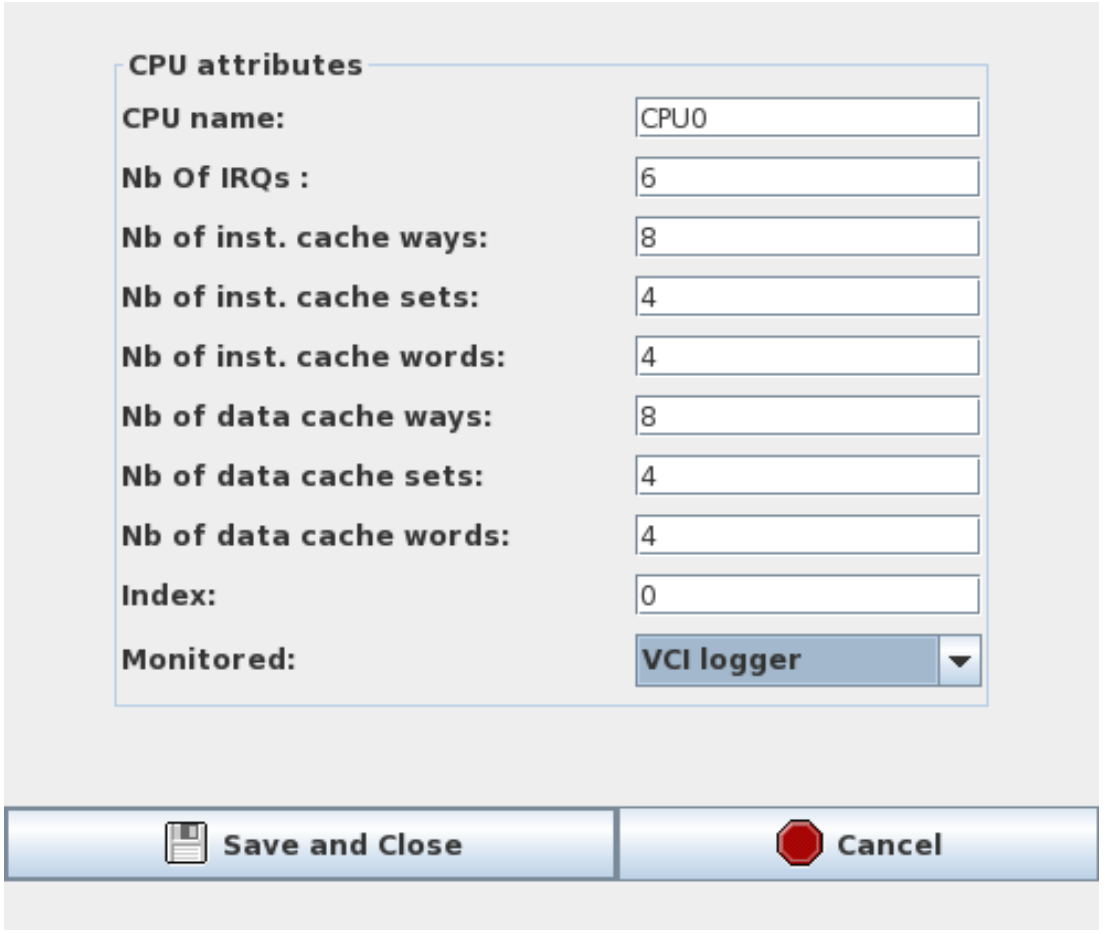
Deployment View



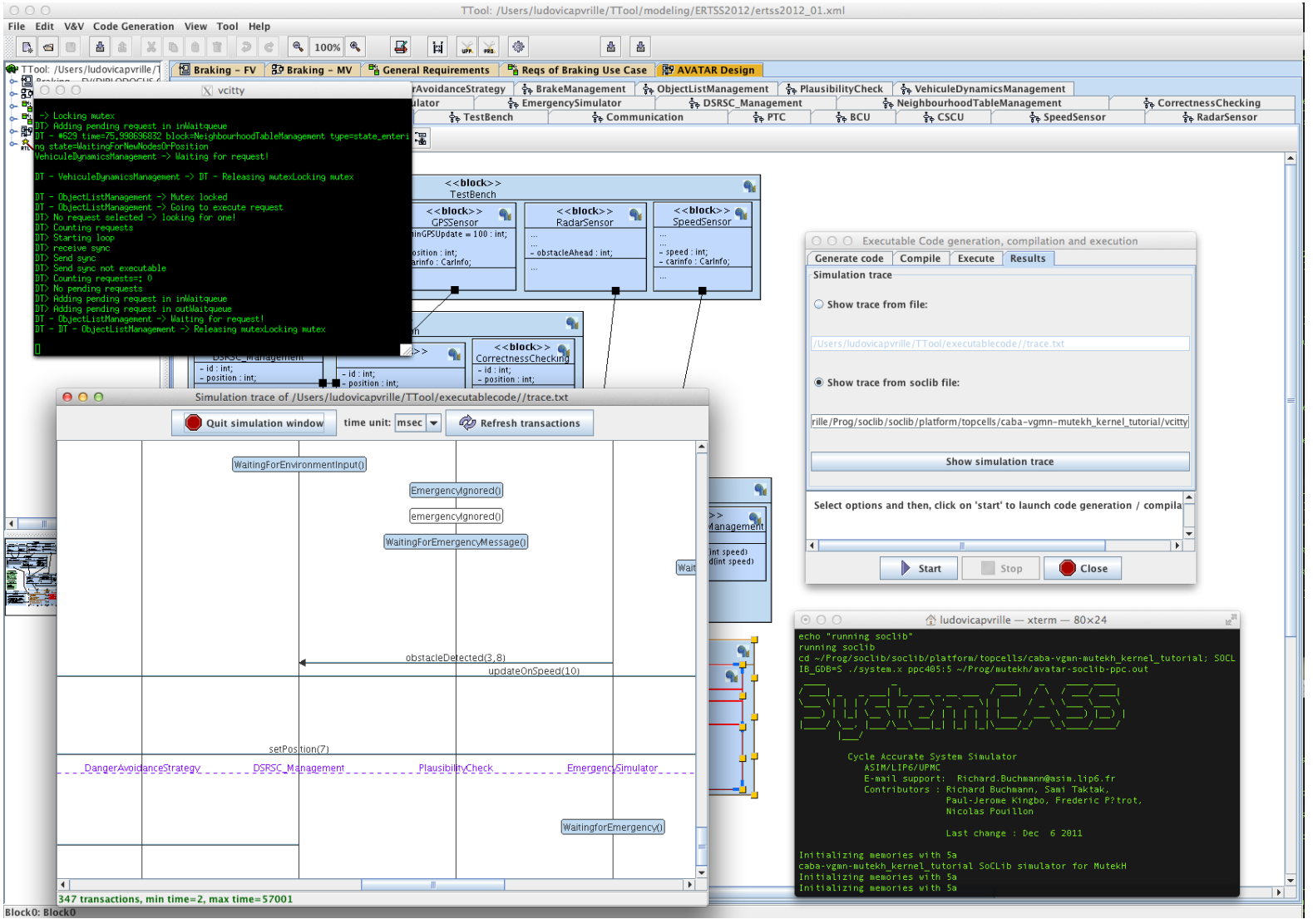
Deployment Diagram

- Tasks mapped to model of target system
- Generating software elements (tasks, main program)
- Hardware elements built from the deployment information

Software Design View: CPU configuration

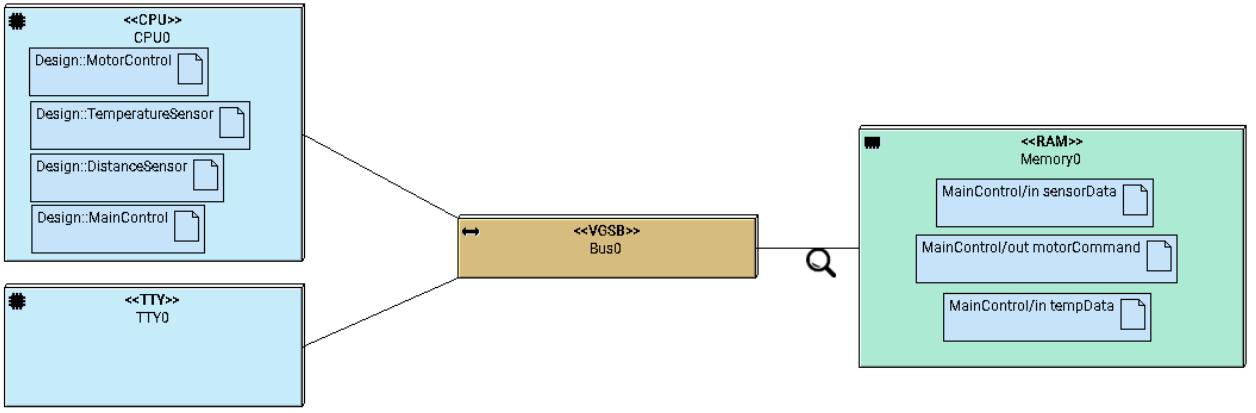


SoCLib Virtual Prototype



Capturing Performance Information

- Logging probes ("spies") can be set on interfaces between interconnect, CPU and memories
- SoCLib allows cycle precise logging
- Python-based scripts generate curves



Typical Performance Information

Partitioning Level: Global Assumptions

- Overhead due to context switching (Cycles per Instruction)
- Memory access latency
  - Data and instruction cache miss
- Bus and NoC contention
- Buffer overflow/underflow

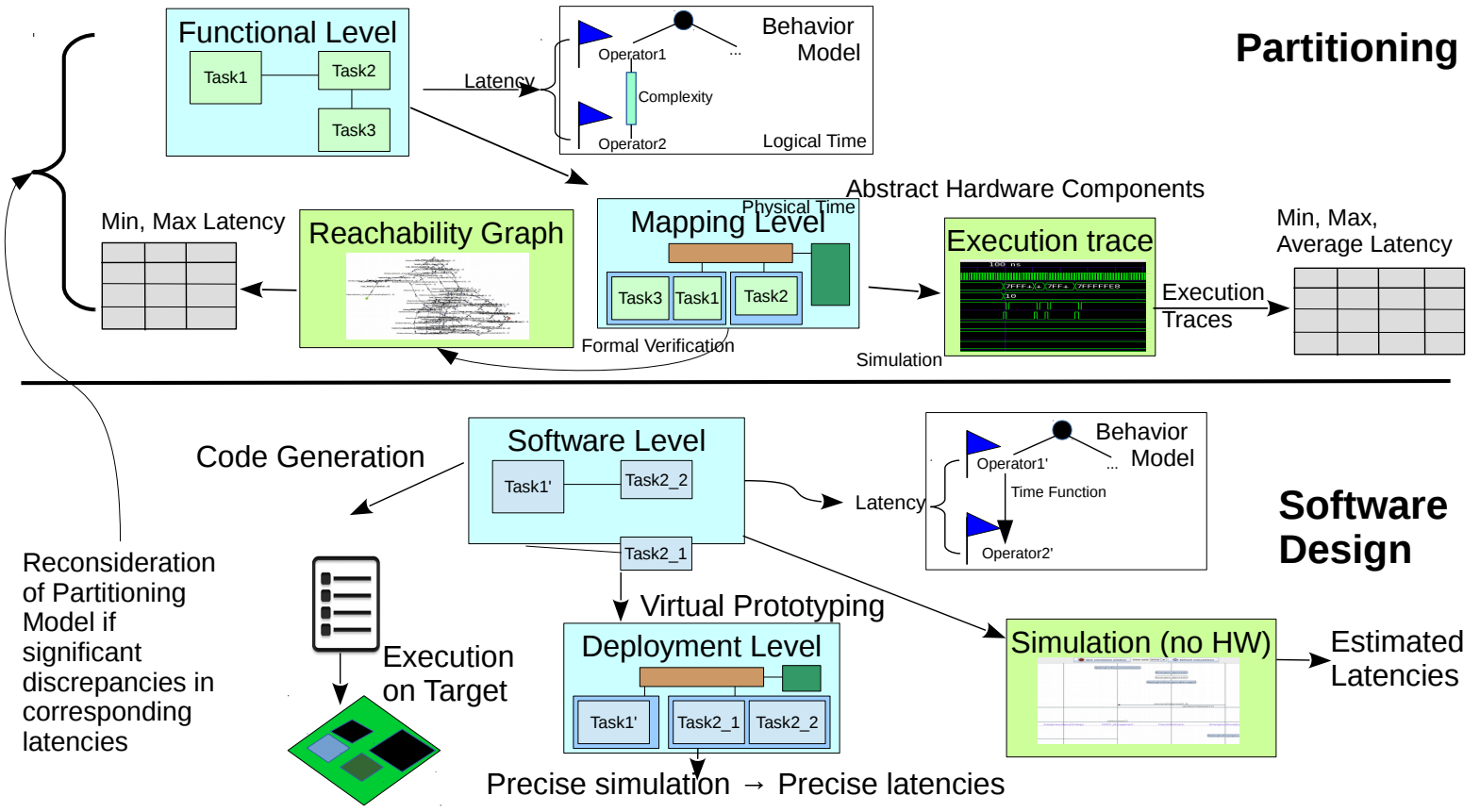
Software Design Level & Virtual Prototype

Confirm or correct assumptions made on higher level

Partitioning and Software Design Level

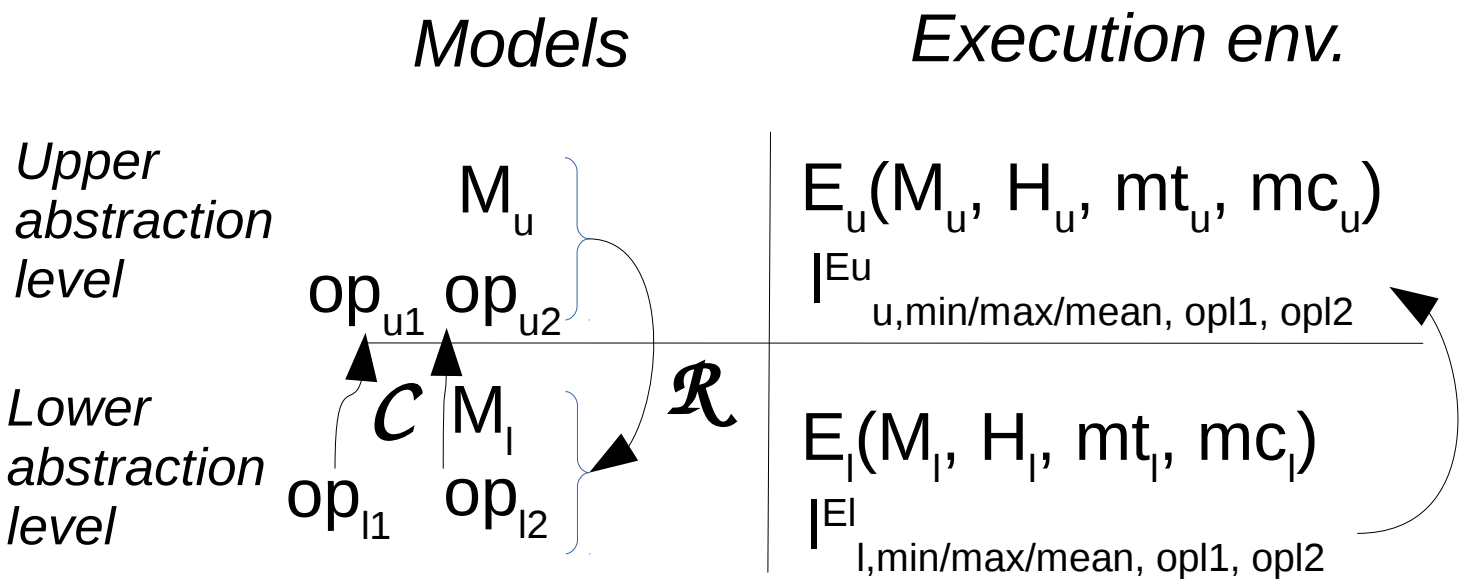
- Latencies: establish formal relation between levels

Overall Method

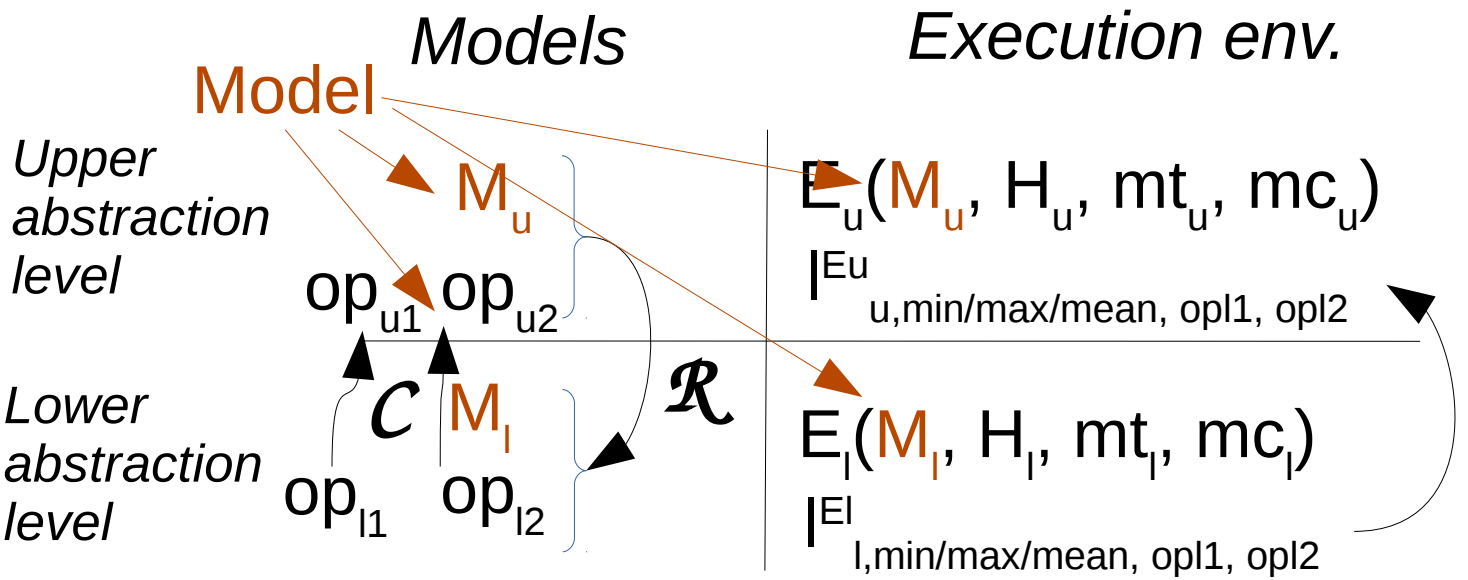


Latency Handling through the Embedded System Design Process

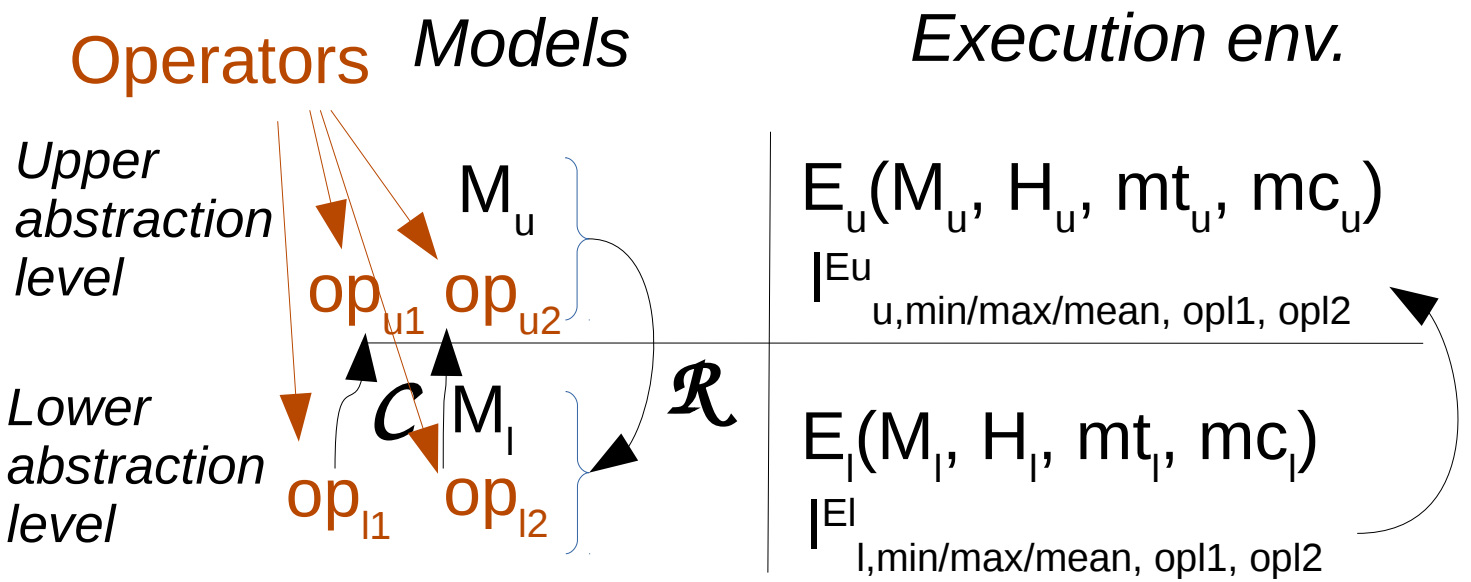
Abstraction Levels and Latencies



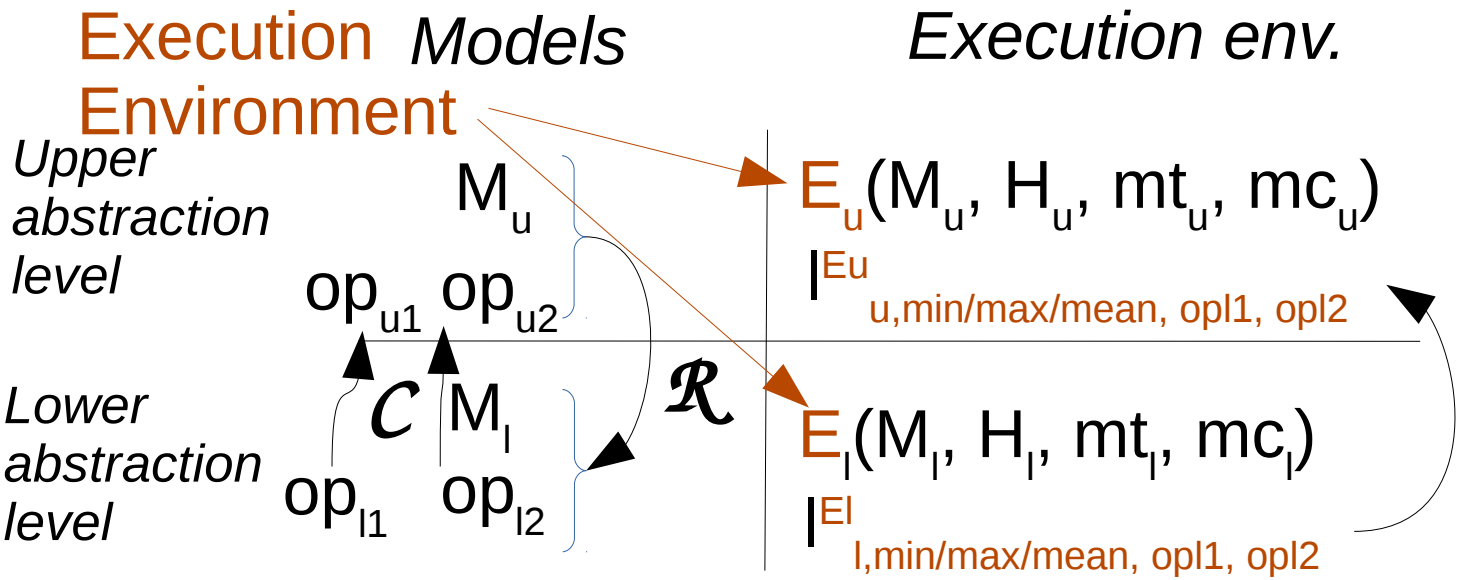
Abstraction Levels and Latencies



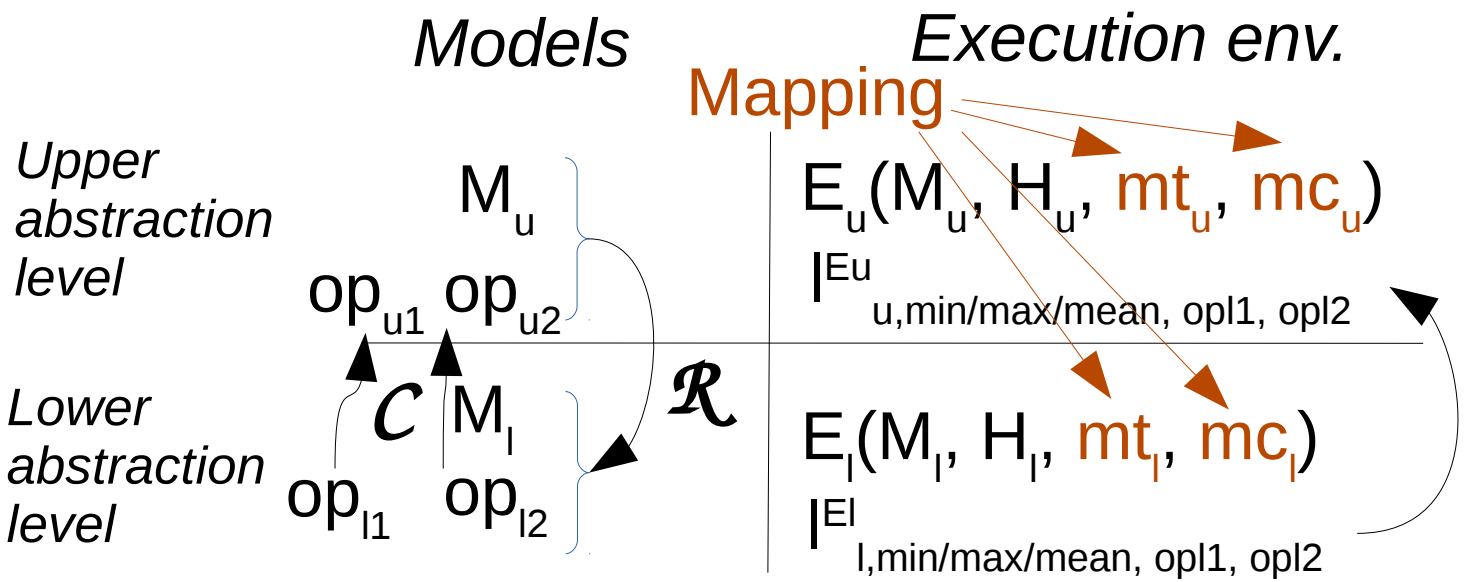
Abstraction Levels and Latencies



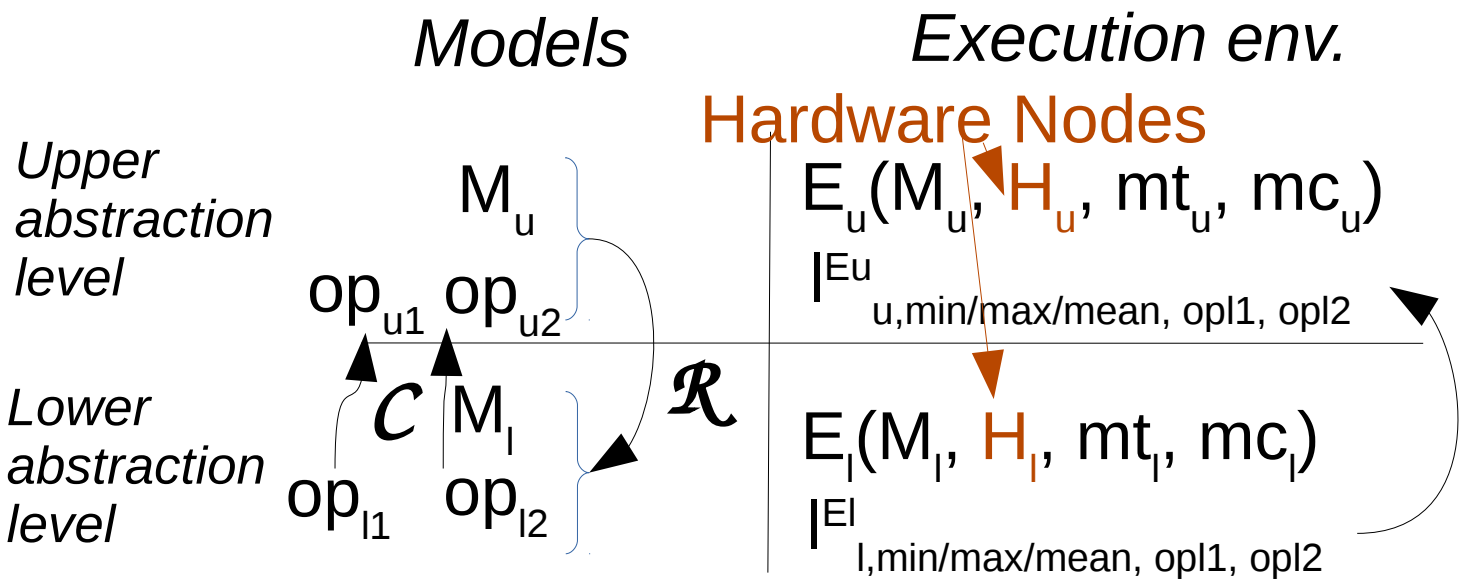
Abstraction Levels and Latencies



Abstraction Levels and Latencies



Abstraction Levels and Latencies



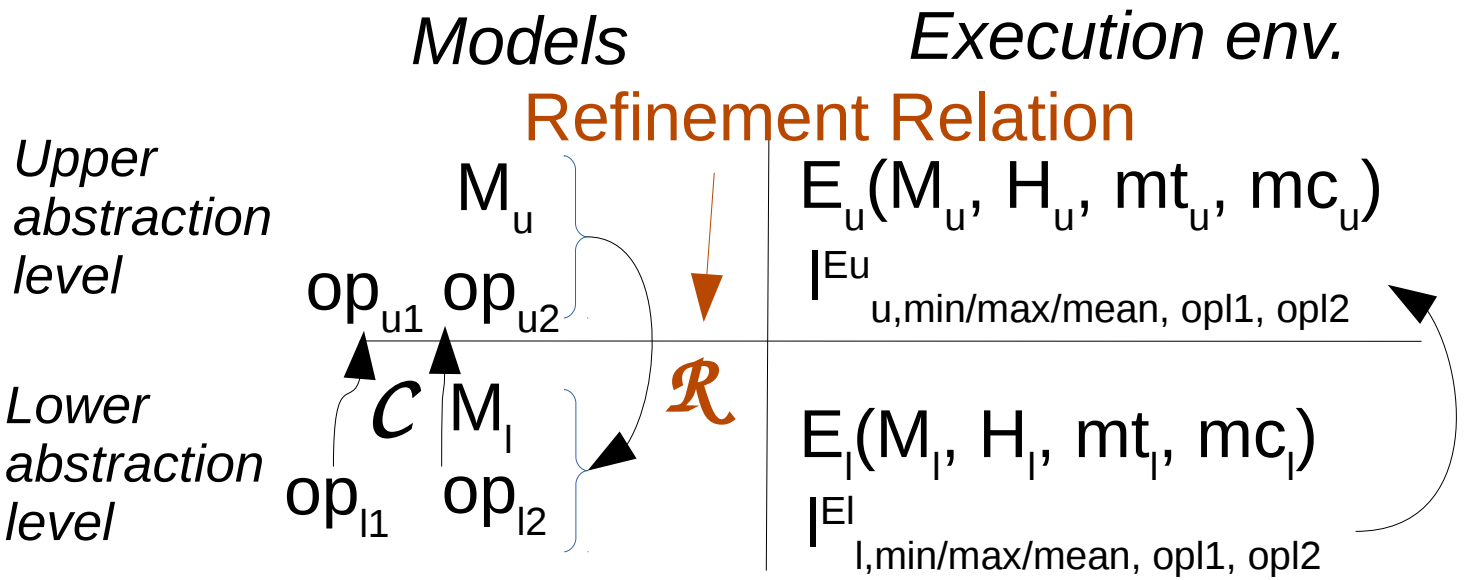
Correspondence between Latencies

- Relate operators of different abstraction levels
- Correspondence relation $\mathcal{C}(op)$ between operators of two abstraction levels u (upper) and l (lower)

$$\forall op^{M_l,t_l}, \text{noted } op_l, \mathcal{C}(op_l) = \begin{cases} op_u \\ \phi \qquad otherwise \end{cases}$$

Latencies can be related when $op_{l,1}$ and $op_{l,2}$ of M_l have both a non empty correspondence in M_u

Refinement Relation $\mathcal{R}$



Partitioning

Functional Level: Behavior

- Control Operators: loops, choices, ...
- Communication Operators: channel and events read/write
- Complexity operators

Latency = logical time between complexity operators

Mapping Level: Architecture Model

- Communication Nodes
- Storage Nodes
- Execution Nodes

Latency = physical time for latencies defined at funct. level

Software Design

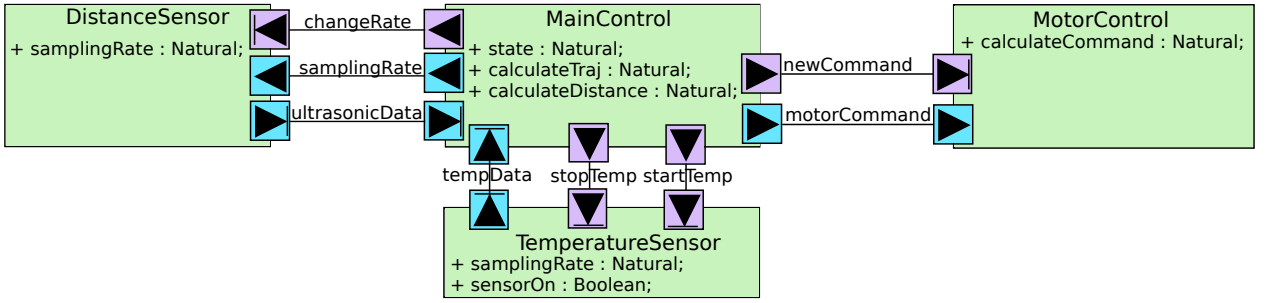
Model transformations realizing the Refinement relation $\mathcal{R}$

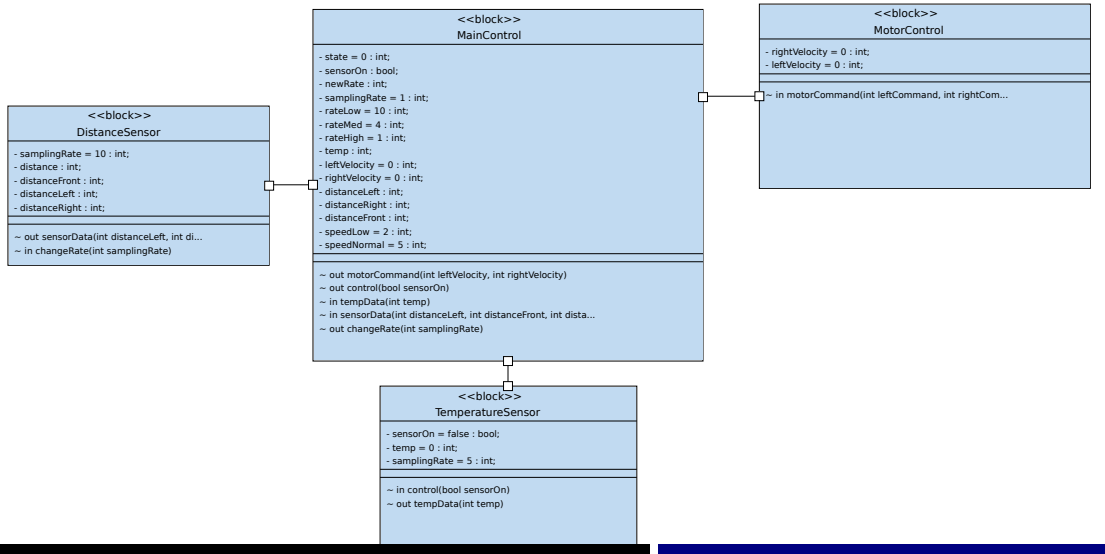
- Tasks can be split into subtasks
- Complexity operators replaced by sub-behavior
- Communications related to split tasks may be added

Significant discrepancies $\rightarrow$ error in one of the models

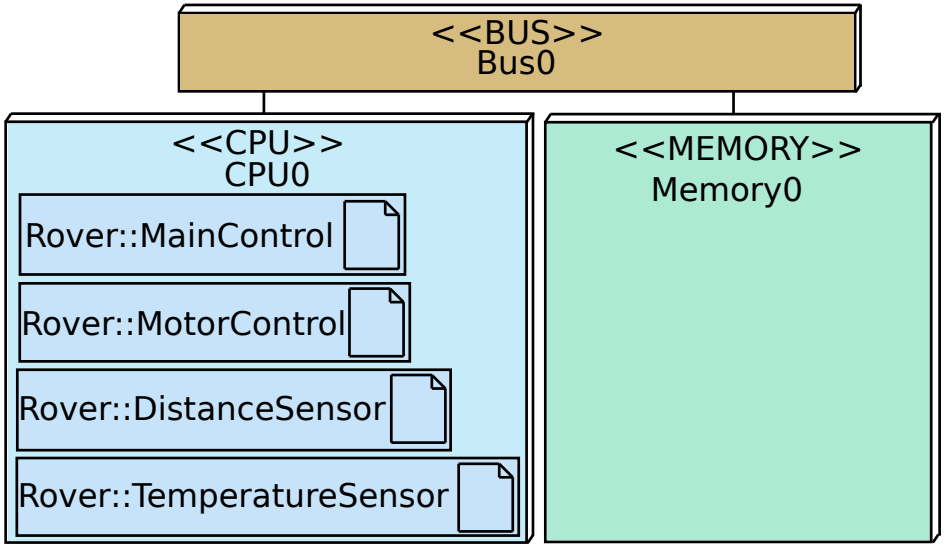
- Estimation for computation complexities for certain algorithms may be wrong
- Inaccurate architecture modeling : cache miss, CPI, etc

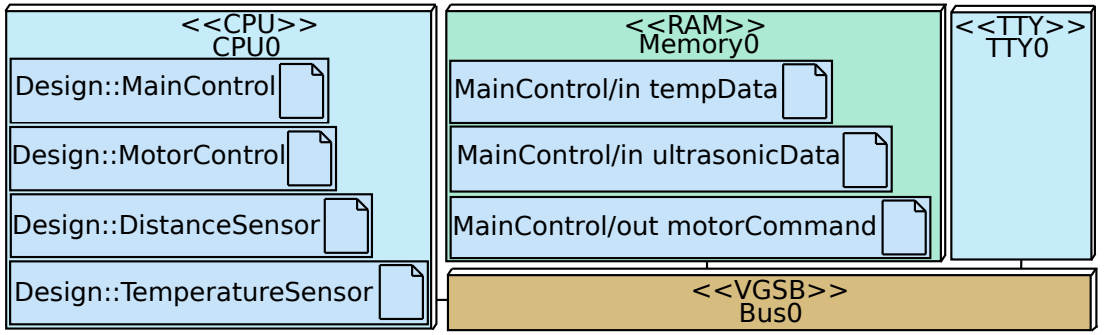
Rover Functional and Software Model



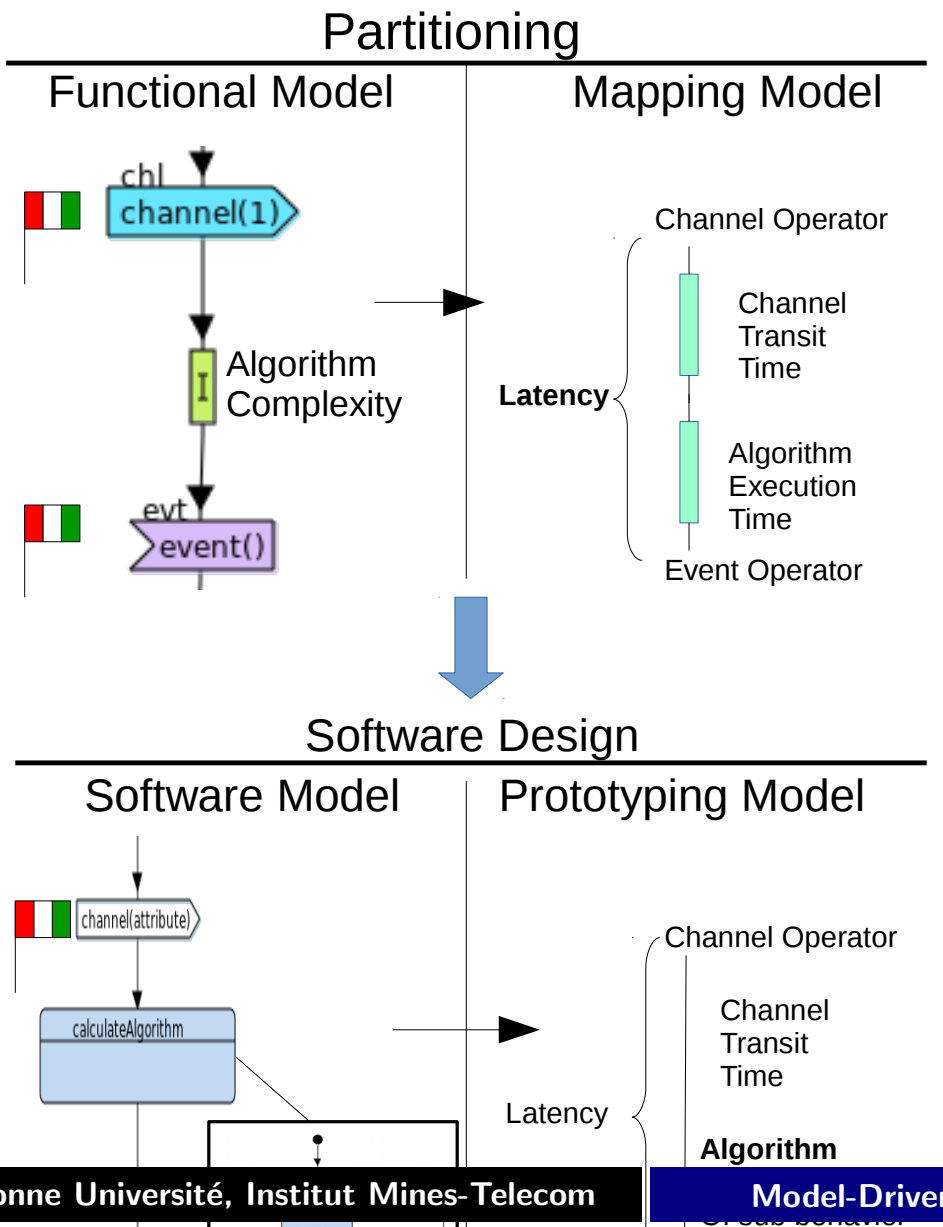


Mapping: Functional and Software Design Level

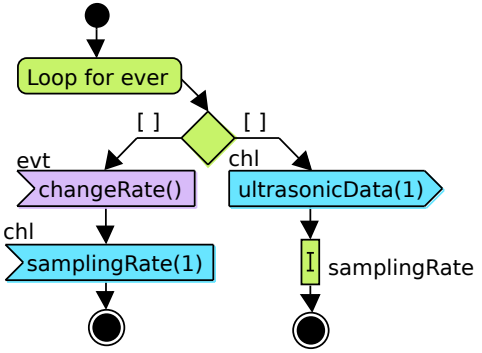


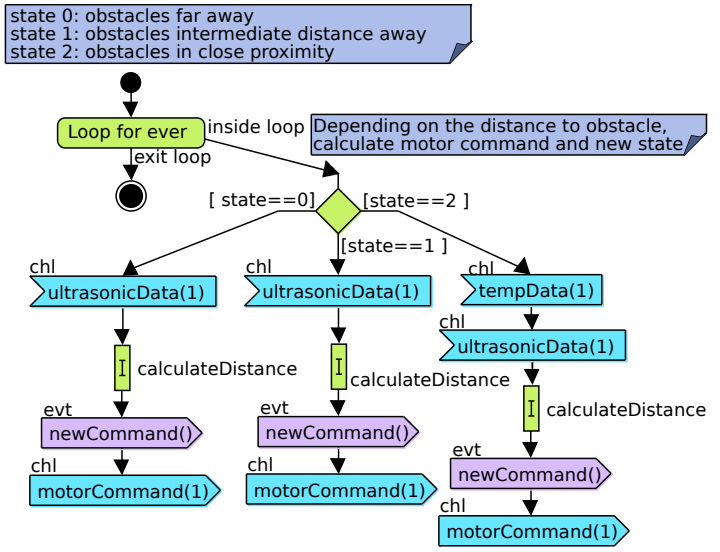


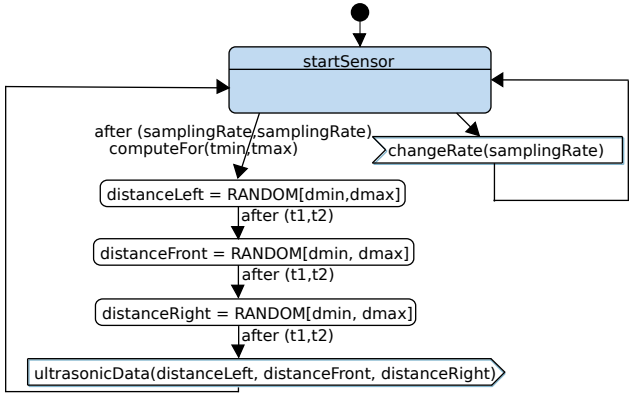
Latencies and Checkpoints in TTool

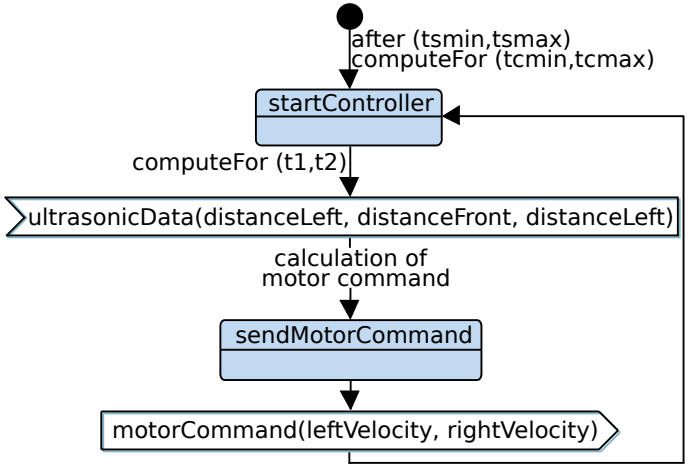


Behaviour Diagrams

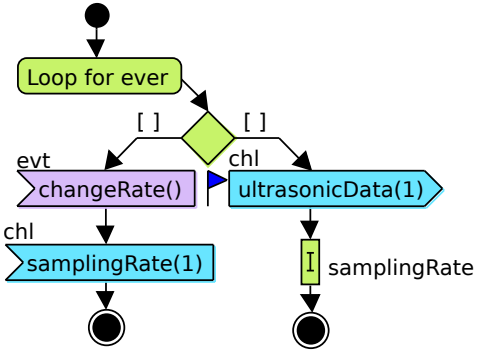


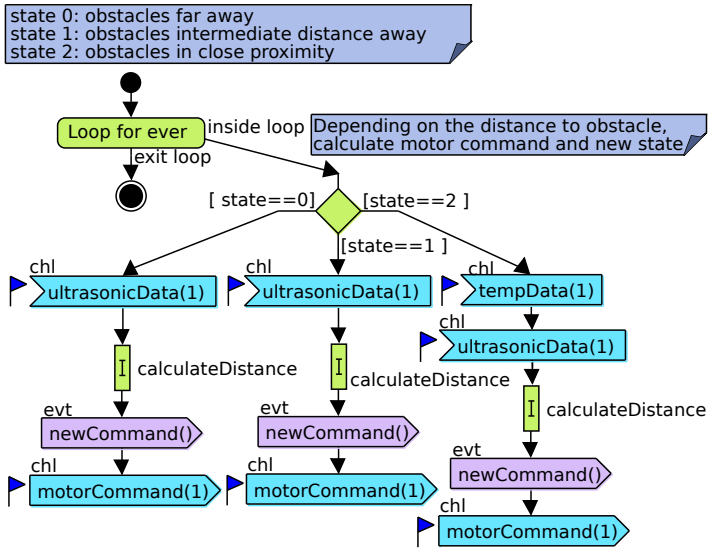


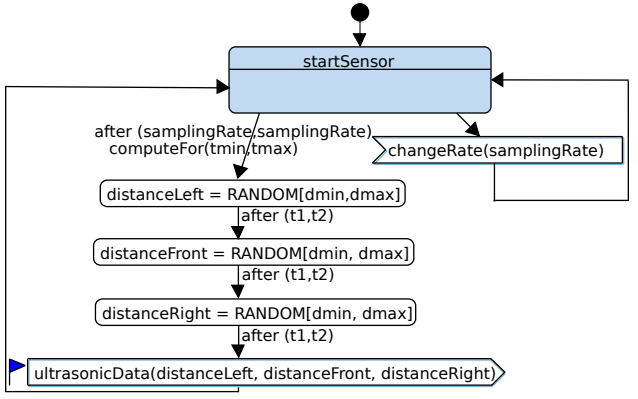


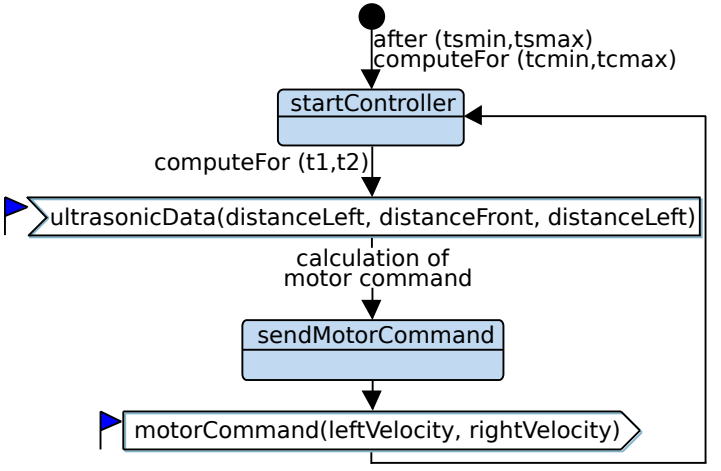


Latency Checkpoints

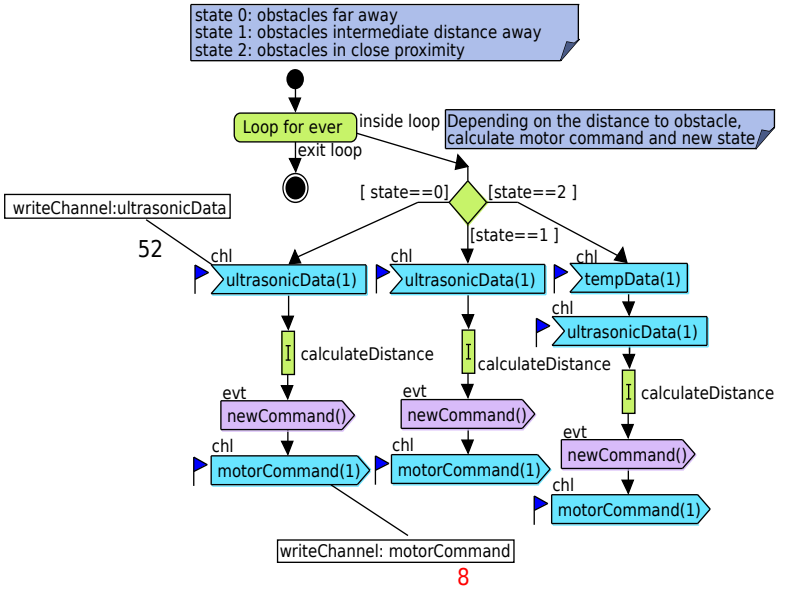


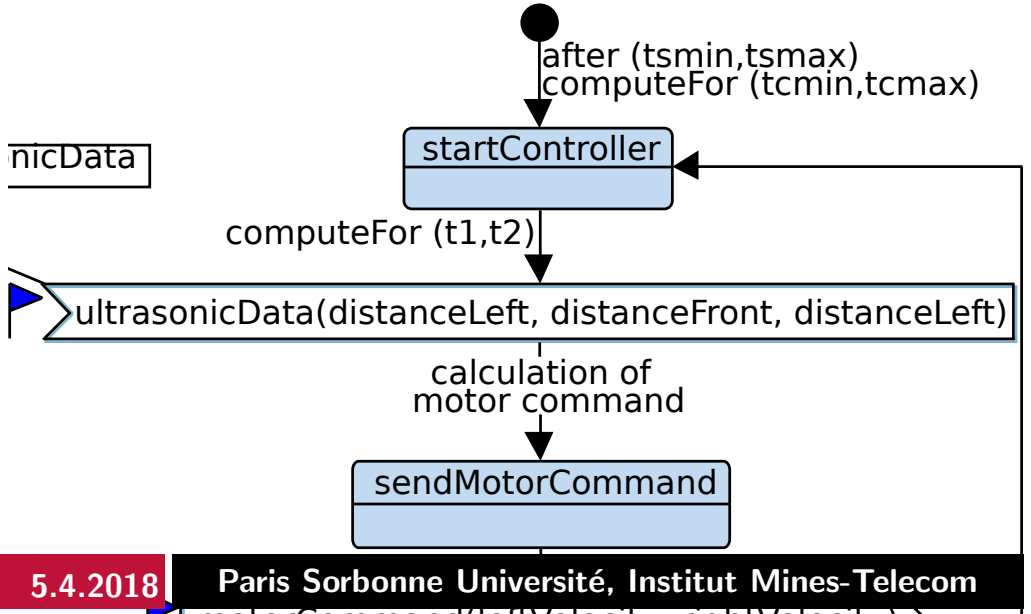






Latency Annotations





Heterogeneous Extension

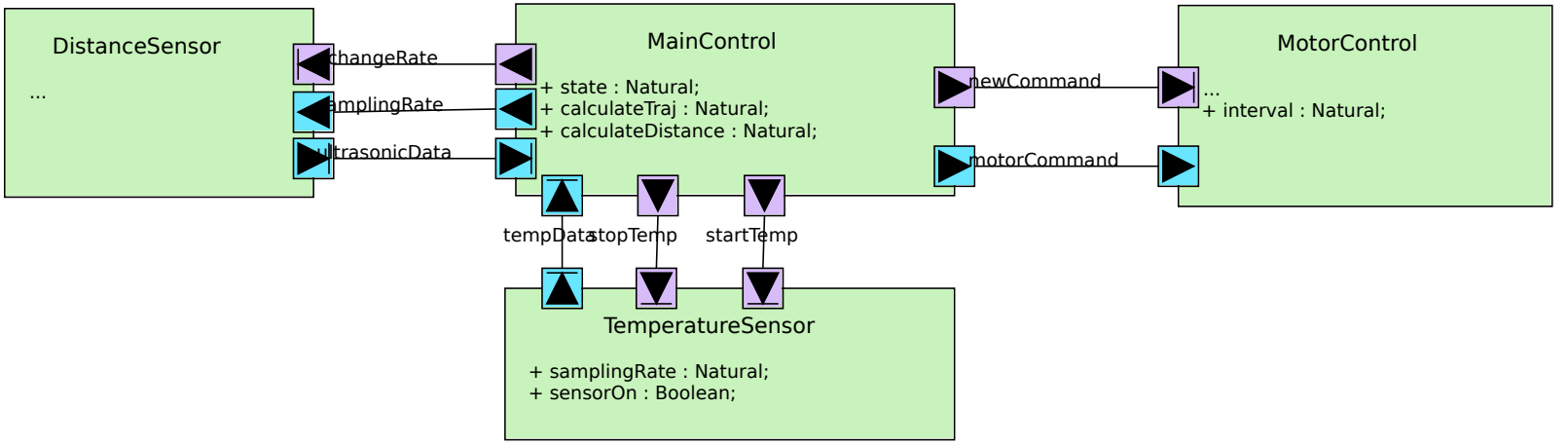
Embedded hardware is increasingly heterogeneous

- Digital/analog integrated circuits, sensors, actuators
- Robotics, automotive, autonomous systems

Heterogeneous Features

- Analog/mixed signal (AMS) features
- Radio frequency (RF) features

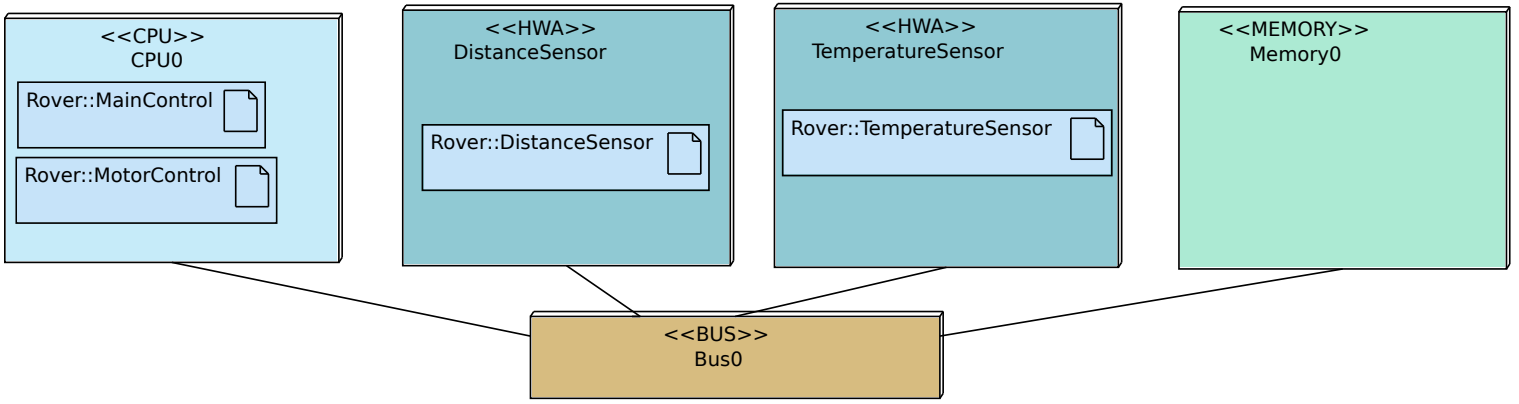
Rover Revisited: Functional Model



⇒ Modeling not realistic

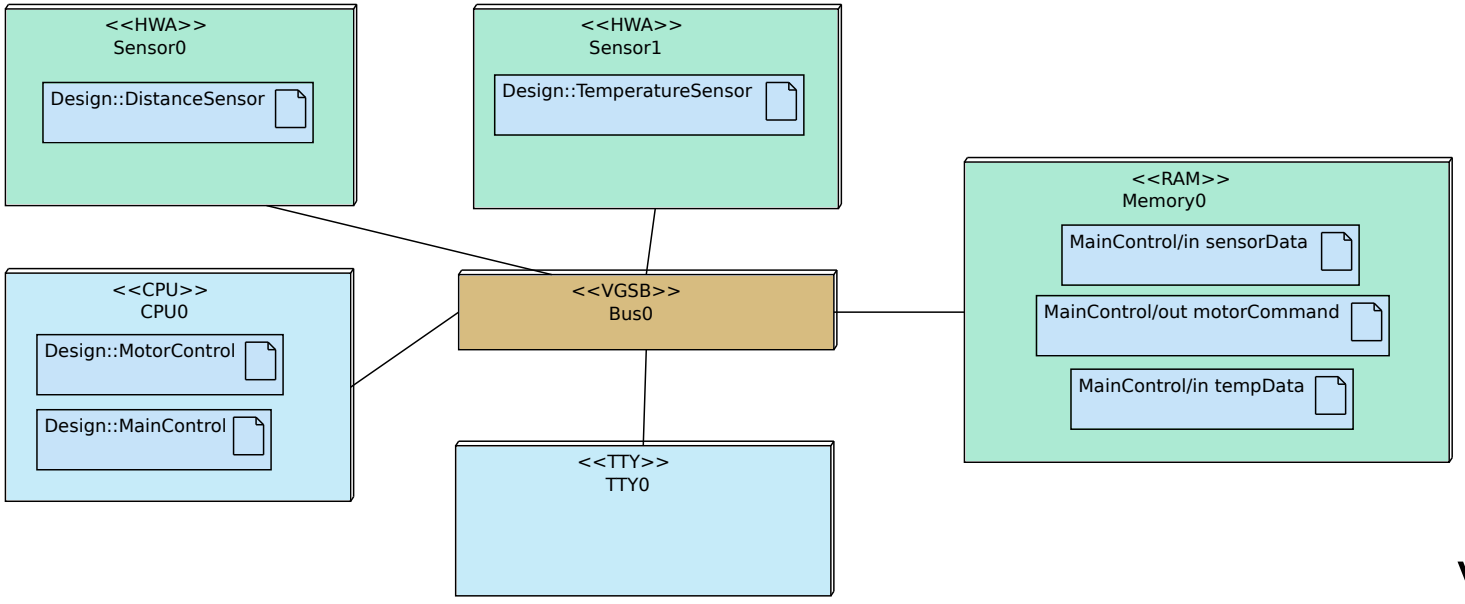
⇒ Requires more adapted representation

Partitioning Level



Hardware/Software partitioning of the rover using the Virtual Coprocessor mechanism of SoCLib

Software Design Level



Virtua

Prototype with (Hardware) Sensors

SystemC AMS Extensions

- Recent standard describing an extension of SystemC with AMS and RF features (2019, 2013 v2.0)
- Modeling formalisms:
  - *Discrete Event* DE
  - *Timed data Flow* TDF
  - *Electrical Linear Networks* ELN
  - Consortium includes Fraunhofer IIS, TU Vienna, EPFL, NXP, Infineon, LIP6

SocLib components can be reused

- Both are based on SystemC
- SystemC is a set of C++ class libraries
- SoCLib digital components are DE modules and can be simulated together with TDF modules

Contribution of 2017/2018 LIP6 Project

First steps towards an integrated modeling and simulation tool for verification and virtual prototyping of heterogeneous embedded systems on different abstraction levels and different models of computation (MoC)

- Methodology to integrate heterogenity into TTool
- SystemC AMS clusters in SysML in TTool panel
- Automated handling of synchronization issues between MoC (L. Andrade 2015, C. BenAoun 2017)
- Code generation for heterogeneous simulation

Related Work

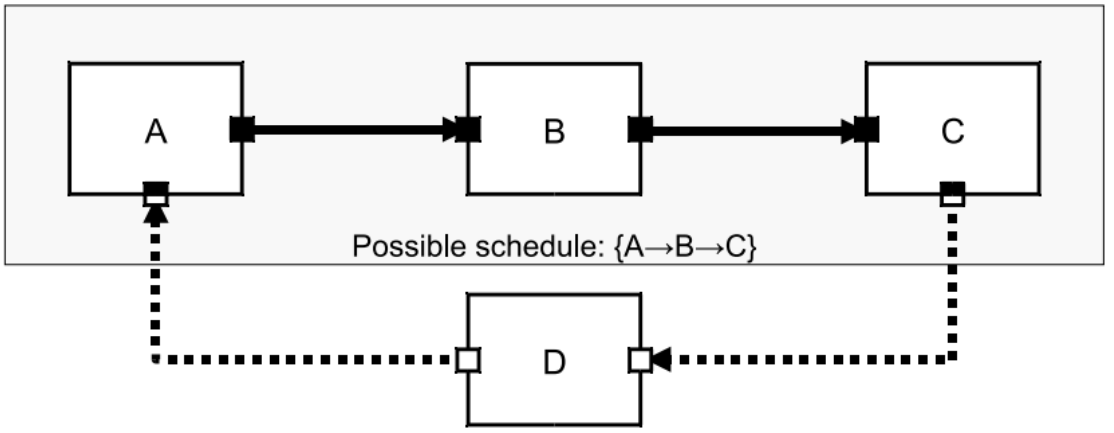
Non SystemC based

- Ptolemy II (Ptolemy.org 2014)
- METROPOLIS (Balarin et al. 2003)
- METRO II (Davare et al. 2007)

SystemC based

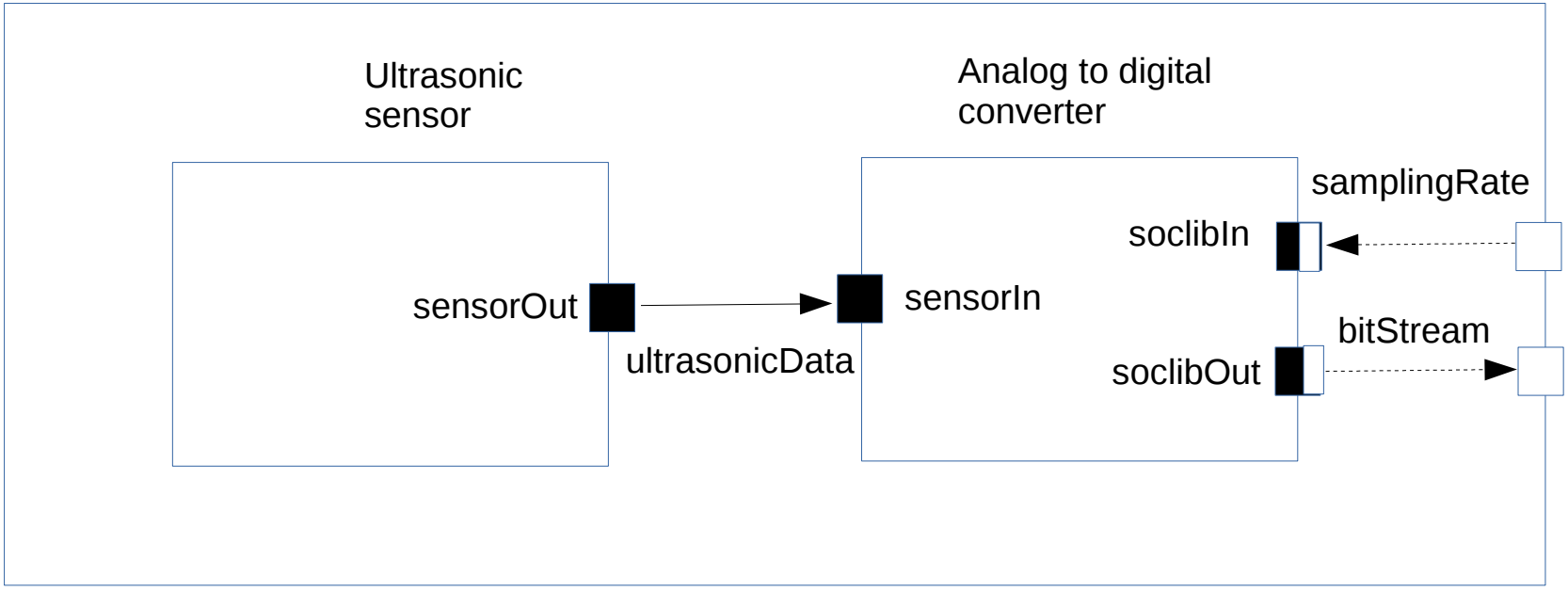
- HetSC (Herrera et al. 2007)
- HetMoC (Zhu et al. 2010)
- ForSyDe (Niaki et al. 2012)

SystemC AMS Extensions



- TDF module
  - time step Tm
  - processing function
- TDF port
  - time step Tp
  - *rate* R
  - *delay* D
- Consistency of time step assignment and propagation
  - $Tm = Tp_{in} * R_{in} = Tp_{out} * R_{out}$

SystemC AMS Extensions



SystemC AMS representation of the distance sensor (overview)

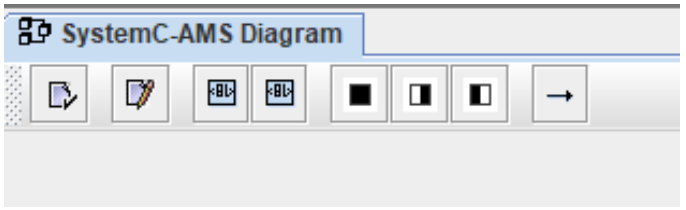
SystemC AMS Representation

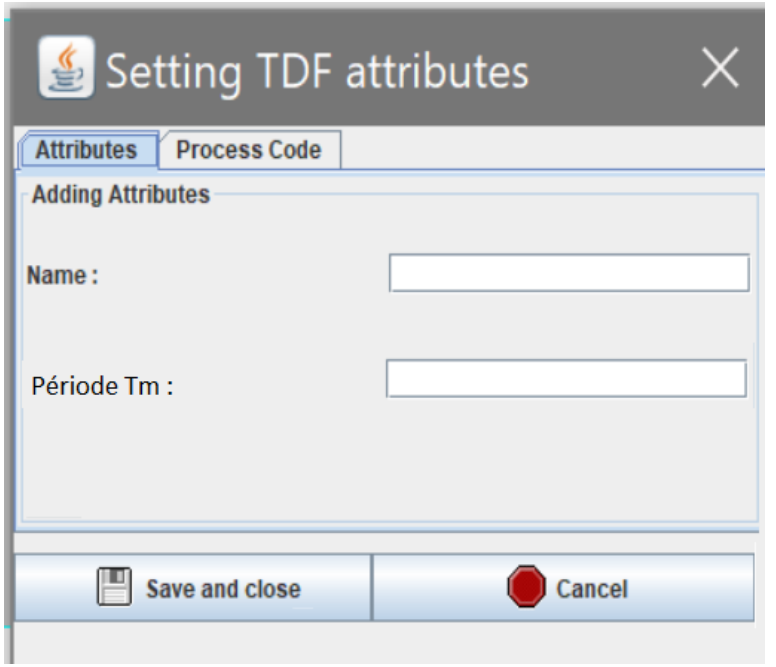
```
template<int NBits>
class adconverter : public sca_tdf::sca_module {

public:
typedef sc_dt::sc_int<NBits> data_type;

sca_tdf::sca_de::sca_in<double> sensorIn;
sca_tdf::sca_de::sca_in<data_type> soclibIn;
sca_tdf::sca_de::sca_out<data_type> soclibOut;

converter<NBits>::adconverter(sc_core::sc_module_name nm)
: sensorIn("in"), soclibIn("soclibIn"), soclibOut("soclibOut") {}

protected:
void set_attributes() {
  sensorIn.set_rate(1);
  sensorIn.set_delay(0);
  soclibIn.set_rate(1);
  soclibIn.set_delay(0);
  soclibOut.set_rate(1);
  soclibOut.set_delay(0);
}
```

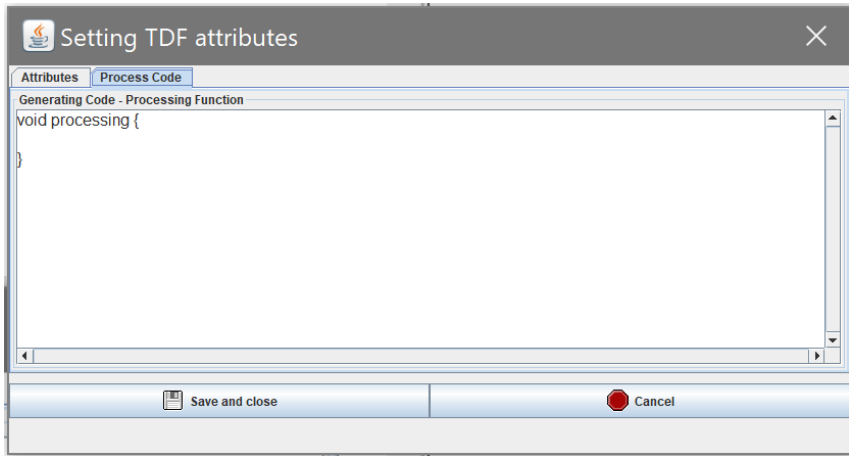
SystemC AMS Representation (2)

```
void processing() {
  data_type res;
  int s =  soclibIn.read();
  int step;
  using namespace std;
  for(step=0; step < s; step++){
    double in = sensorIn.read();
    }
  sc_dt::sc_int<NBits> res =
    lround((in/maxVoltage)*((1<<(NBits-1))-1));
  soclibOut.write(res);
  }
}
```

SystemC AMS Panel Elements for TTool







Conclusion

- Model-driven environment for handling performance issues at different abstraction levels
- Backtracing of performance information to higher abstraction levels
- Integration of system-level design space exploration and prototyping in the same toolkit
- Push-button approach to verification and simulation
- Outline of a method to integrate analog components

Future Work

- Design Space Exploration, with automatically suggested modifications
- More detailed performance profiles (cache misses, processor workload, buffer fill state etc.)
- Library of SystemC-AMS blocks
- Software parts running on (digital) general-purpose processors under a (light) operating system with SystemC-AMS simulation of analog and RF components

References

TTool: ttool.telecom-paristech.fr
SoCLib: www.soclib.fr


Genius, D., Li, L. W., Apvrille, L.: Model-Driven Performance Evaluation and Formal Verification for Multi-level Embedded System Design, Model-Driven Engineering and Software Development, Porto, Portugal (bes paper award, 2017)


Genius, D., Li, L. W., Apvrille, L.: Multi-level Latency Evaluation with an MDE Approach, Model-Driven Engineering and Software Development, Funchal, Portugal (2018)


D. Genius, M.-M. Louërat, F. Pêcheux, L. Apvrille, H. Stratigopoulous: odeling Heterogeneous Embedded Systems with TTool, 5th Workshop on Design Automation for Understanding Hardware Designs (DUHDe), Dresden, Germany (2018)