



An ADL centric approach for the formal design of real-time systems

Sébastien Faucou, Anne-Marie Déplanche and Yvon Trinquet

`name@irccyn.ec-nantes.fr`

Institute of Research in Communications and Cybernetics of Nantes
(UMR n° 6597 CNRS / ECN, EMN, University of Nantes)

Real-Time Systems group



└ Context: the REACT project

- ▶ an ADL (CLARA)
- ▶ + some formal analysis tools and techniques
- ▶ + some architecture transformation/mapping algorithms
- ▶ + a rigorous design process

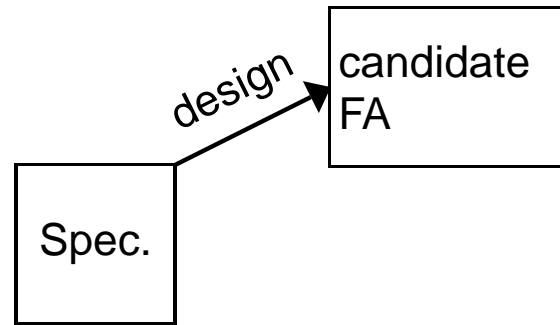
= **REACT**: REal-time Application Configuration Tool

Our goal: seamless integration of formal analysis and implementation techniques into an architecture-based design process

└ Design process

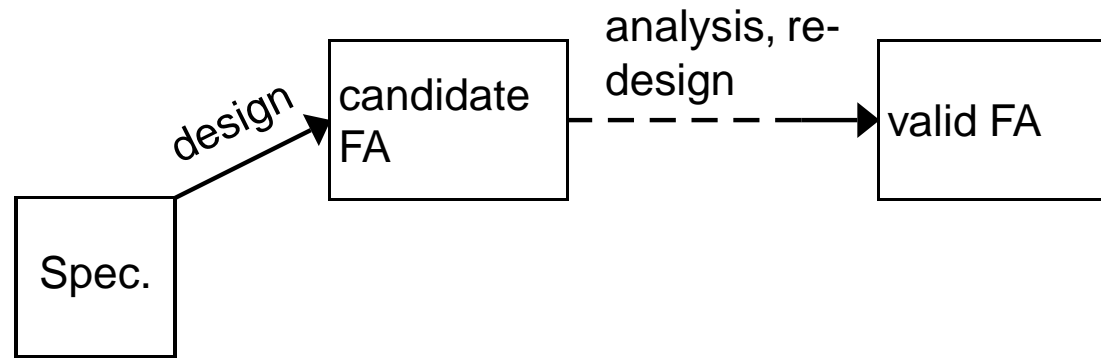
Spec.

Design process



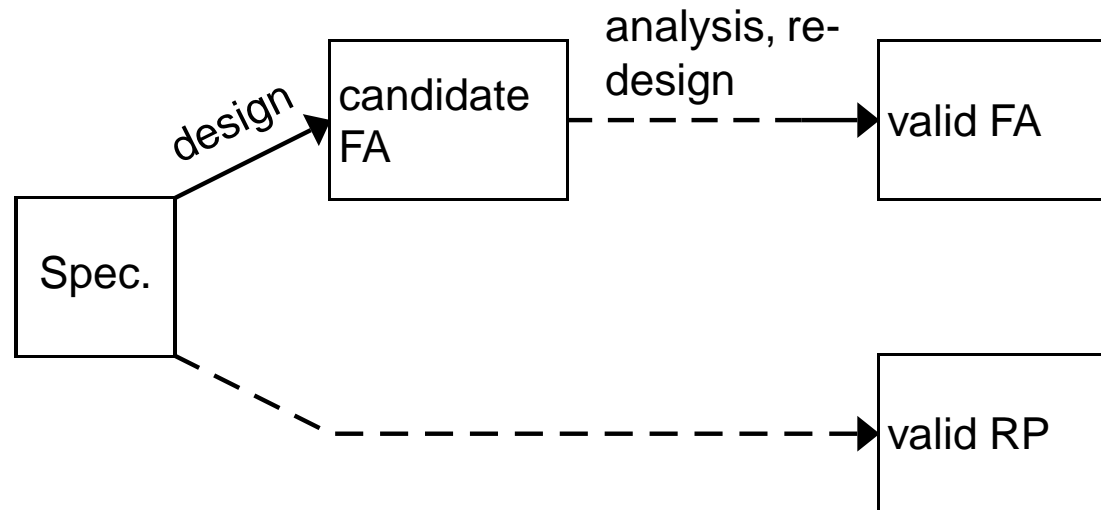
- ▶ FA: functional architecture

Design process



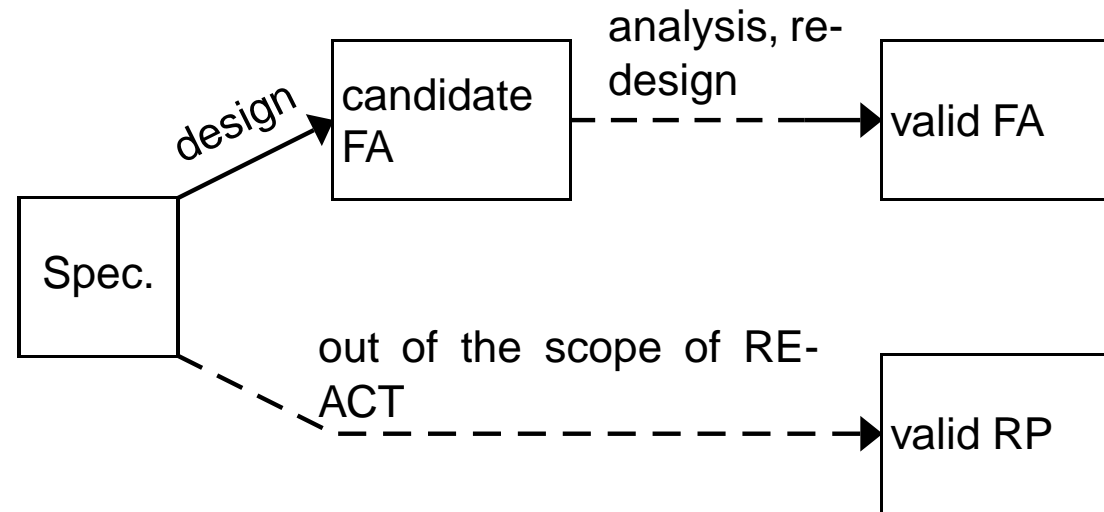
► FA: functional architecture

Design process



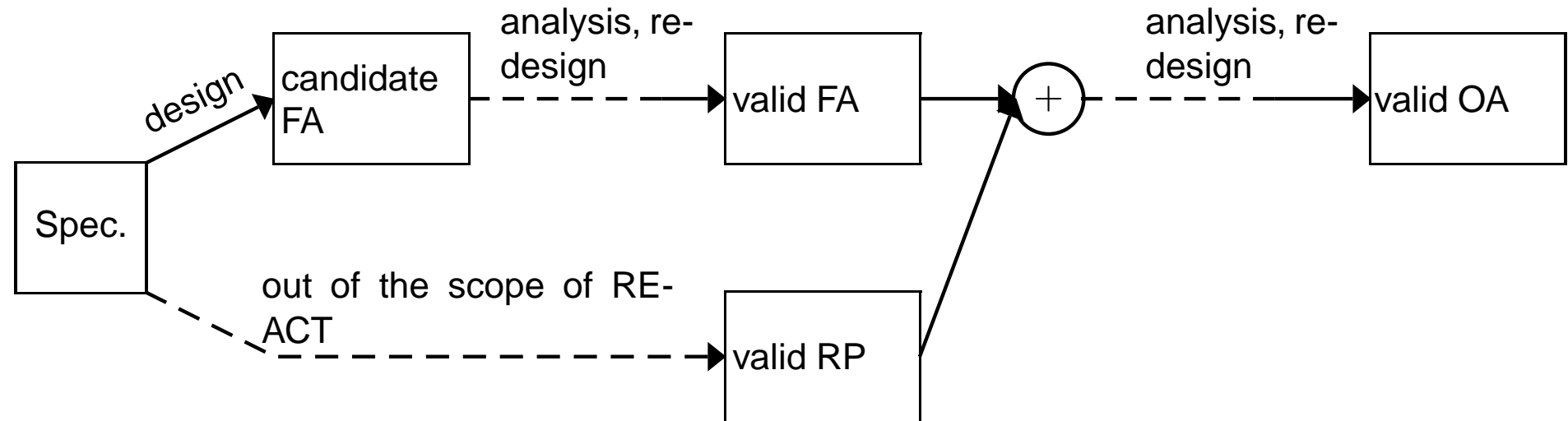
- ▶ FA: functional architecture
- ▶ RP: runtime platform (Hw + low-level Sw)

Design process



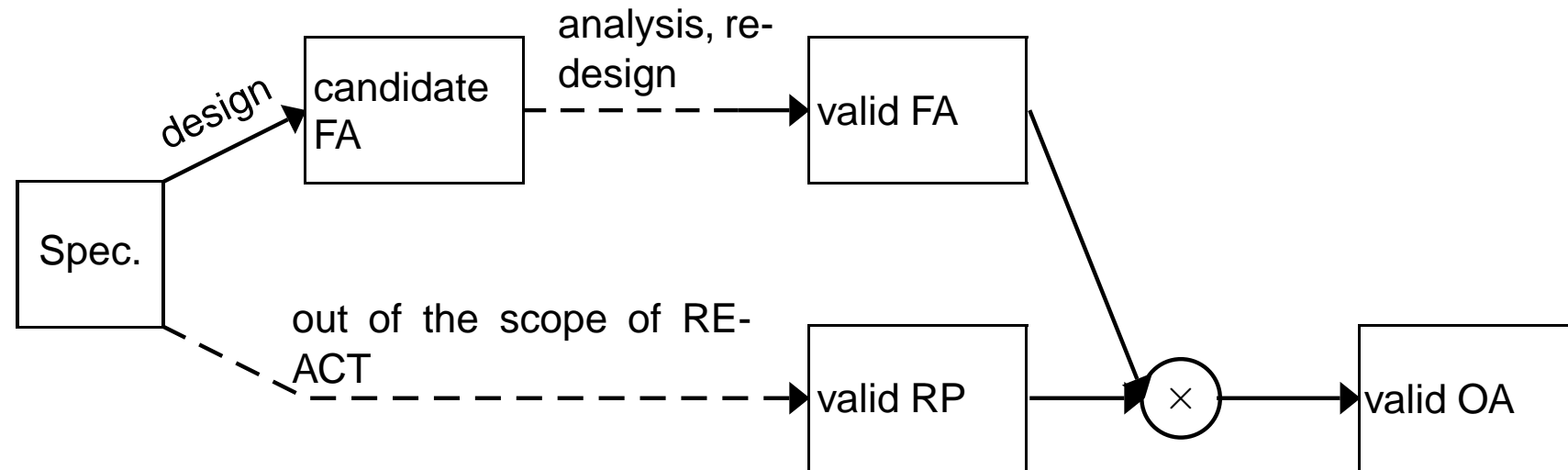
- ▶ FA: functional architecture
- ▶ RP: runtime platform (Hw + low-level Sw)

Design process



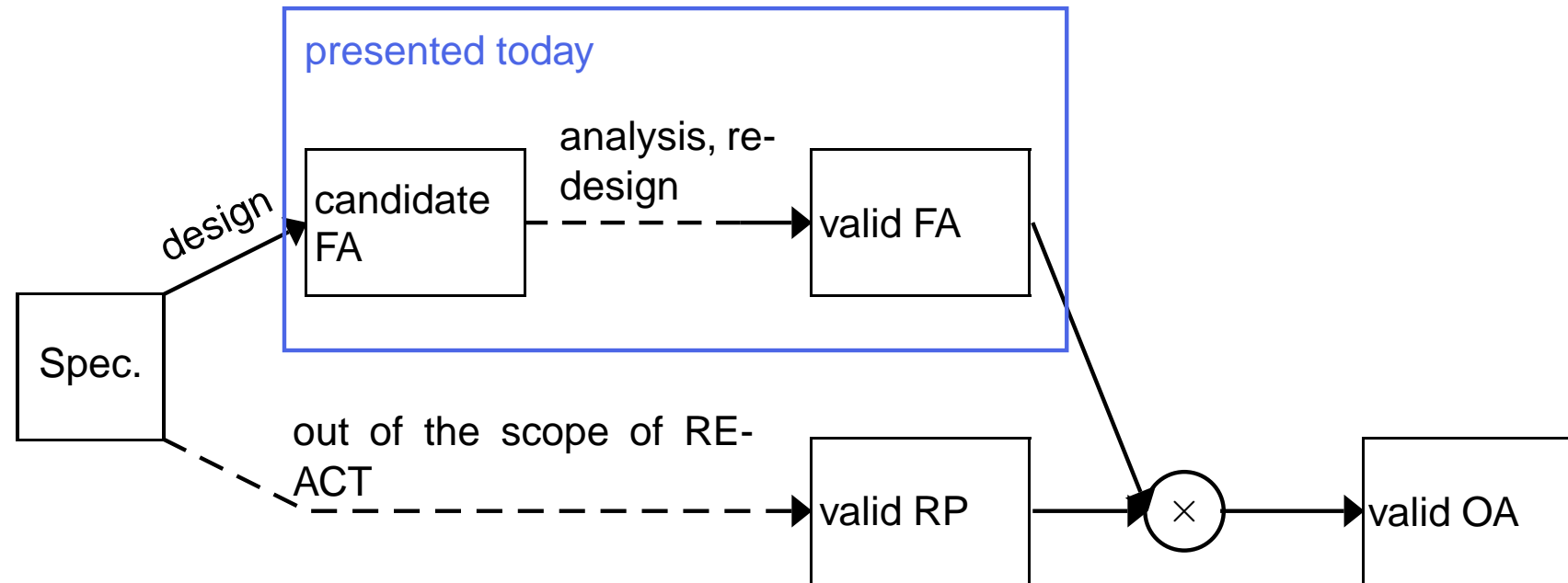
- ▶ FA: functional architecture
- ▶ RP: runtime platform (Hw + low-level Sw)
- ▶ OA: operational architecture

Design process



- ▶ FA: functional architecture
- ▶ RP: runtime platform (Hw + low-level Sw)
- ▶ OA: operational architecture

Design process



- ▶ FA: functional architecture
- ▶ RP: runtime platform (Hw + low-level Sw)
- ▶ OA: operational architecture

└ The ADL: CLARA

CLARA: Configuration LAnguage for Real-time Applications

- ▶ description of the **functional architecture**
- ▶ **accurate description of the control flows**
- ▶ support for the description of **complex interaction policies**
- ▶ **real-time**: periodic and aperiodic triggers, time budgets and constraints
- ▶ hierarchical (top-down design approach), graphical, simple (only what we need)
- ▶ **formal methods friendly**

└ An example

Implementation of a (very simple) feedback control loop :

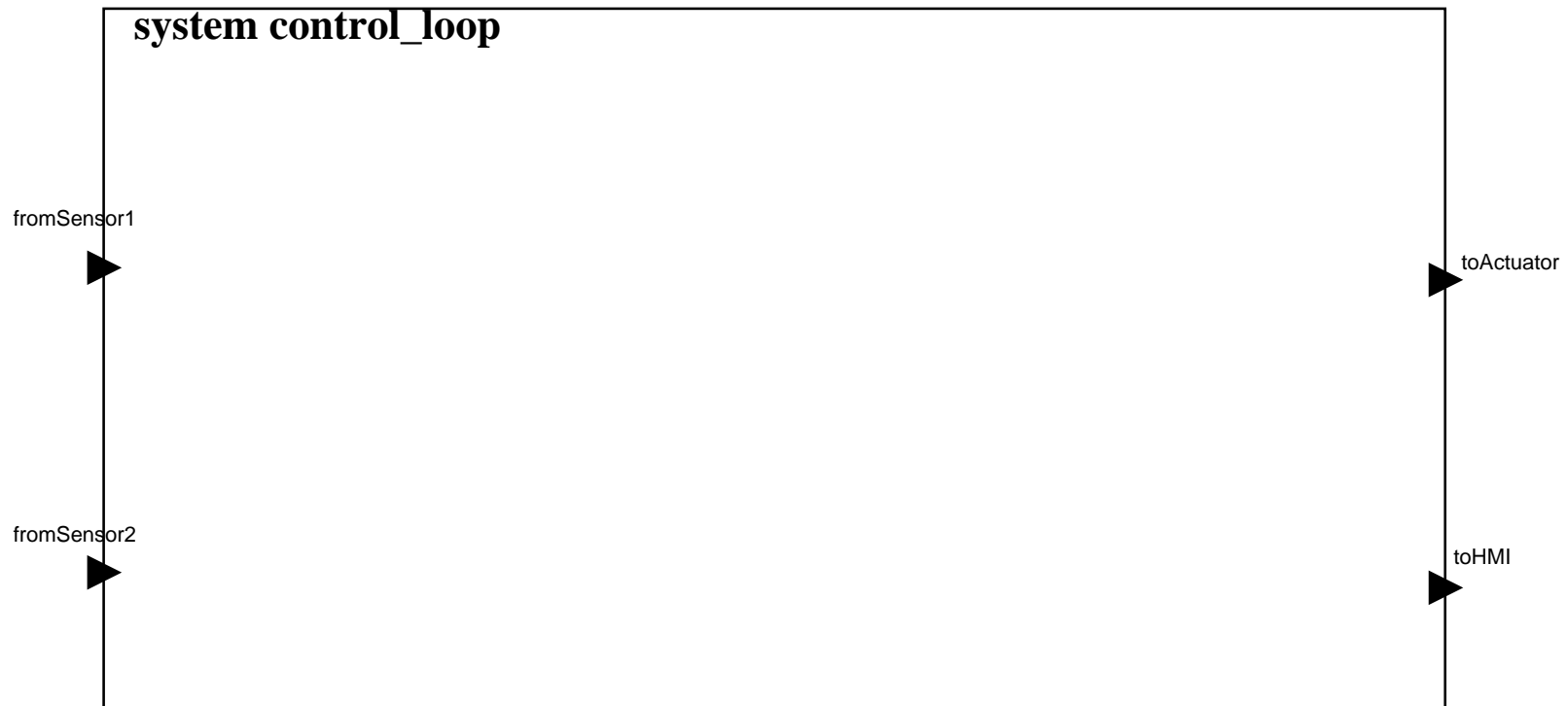
Functional requirements:

- ▶ f1: depending on the controlled variable value, compute and set the value of the manipulated variable. The value of the controlled variable is not spatially homogeneous (2 values needed).
 - Periodic function, period = 10 ms
 - Hard real-time function, deadline = 10 ms
- ▶ f2: draw and send the state of the controlled process to a display.
 - Periodic function, period = 20 ms
 - Soft real-time function, deadline = 20 ms, accepted miss ratio = 0.5

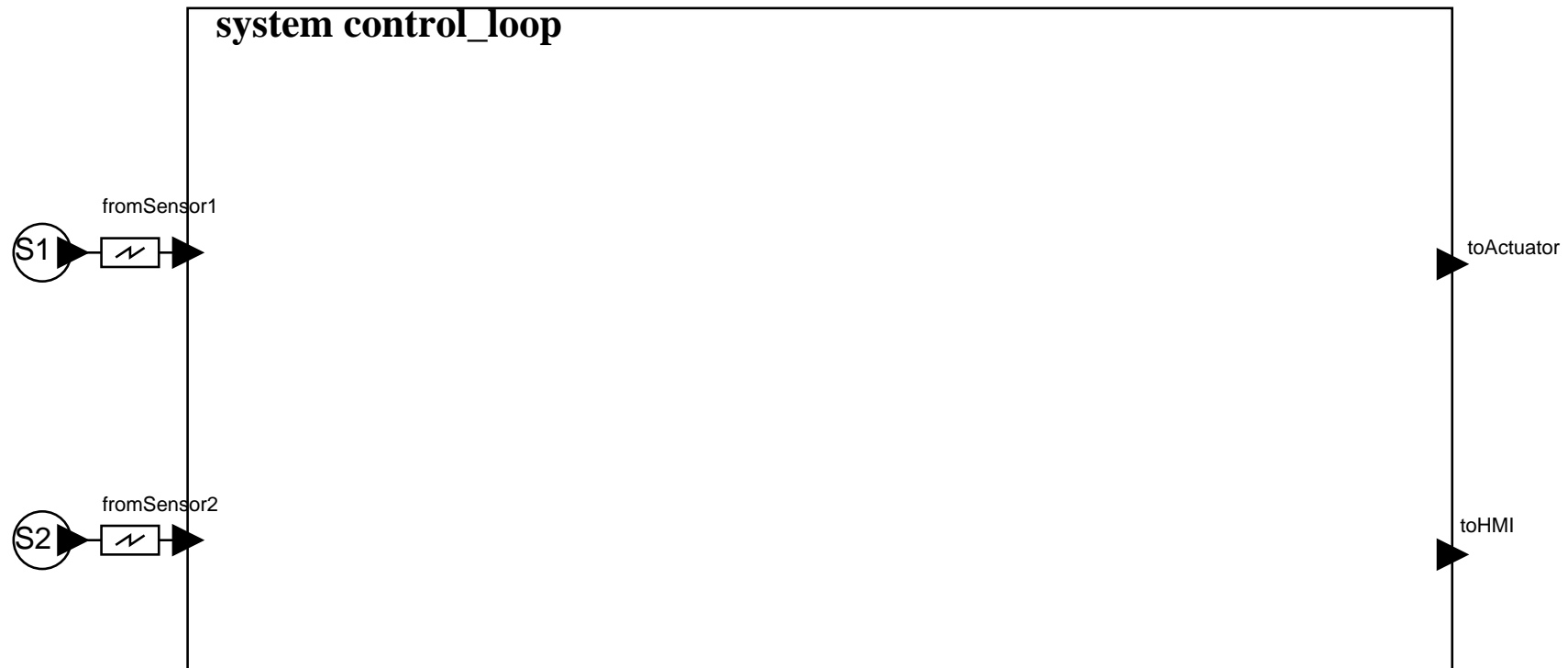
Hardware architecture: 2 sensors, 1 actuator, 1 μ -controller

└ A possible CLARA description

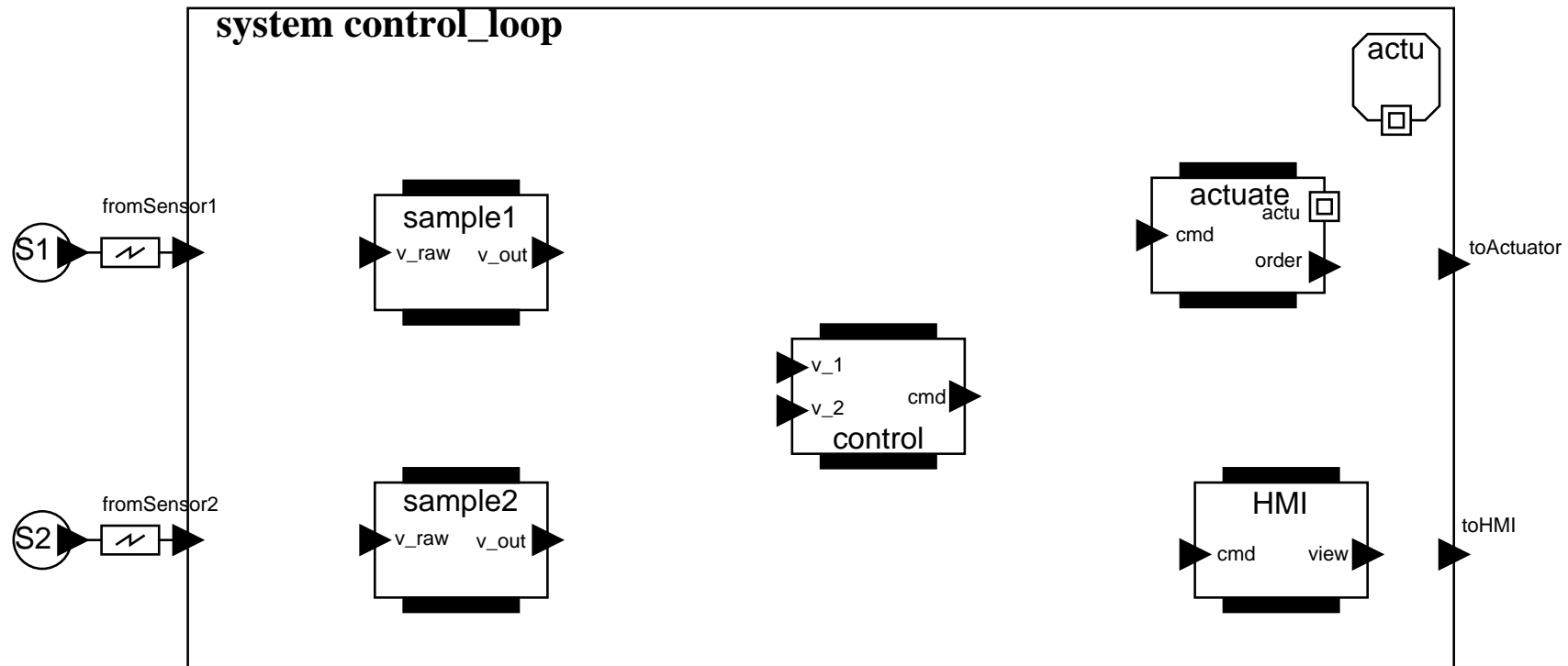
A possible CLARA description



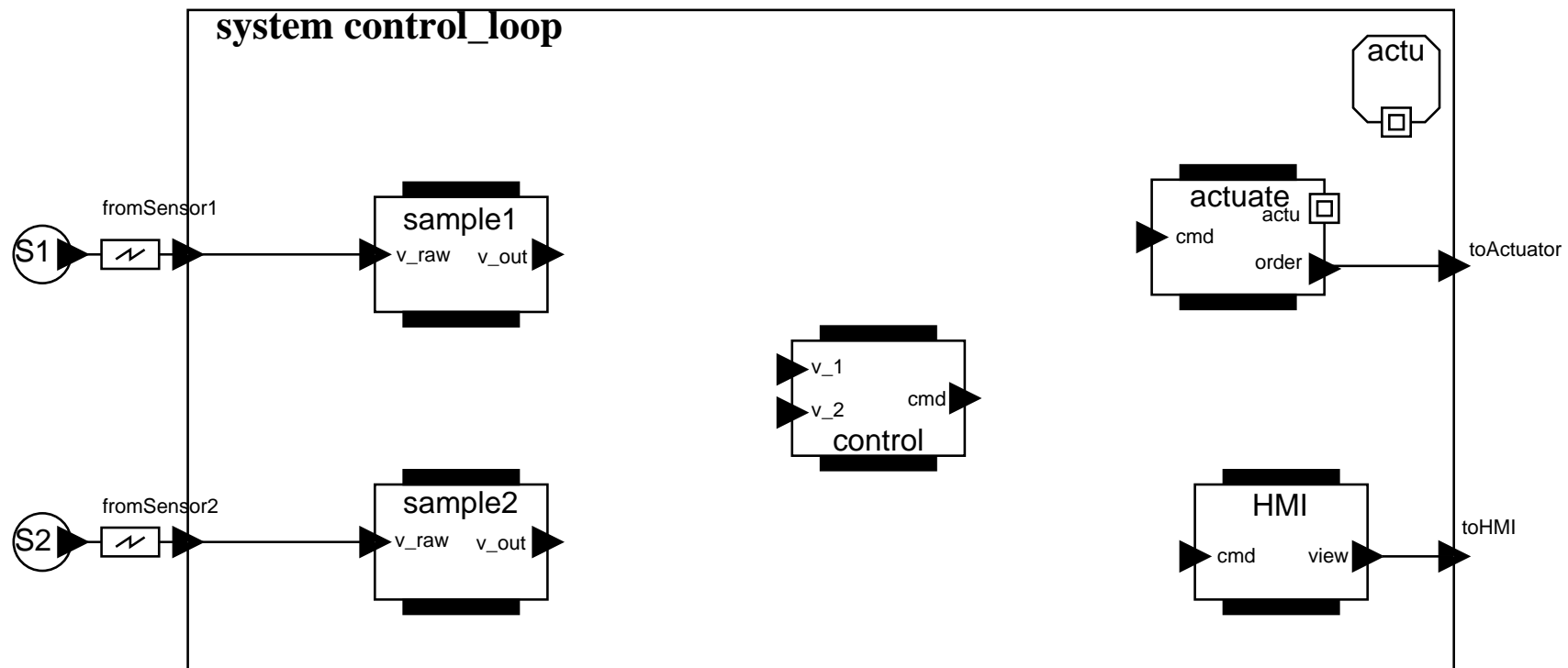
A possible CLARA description



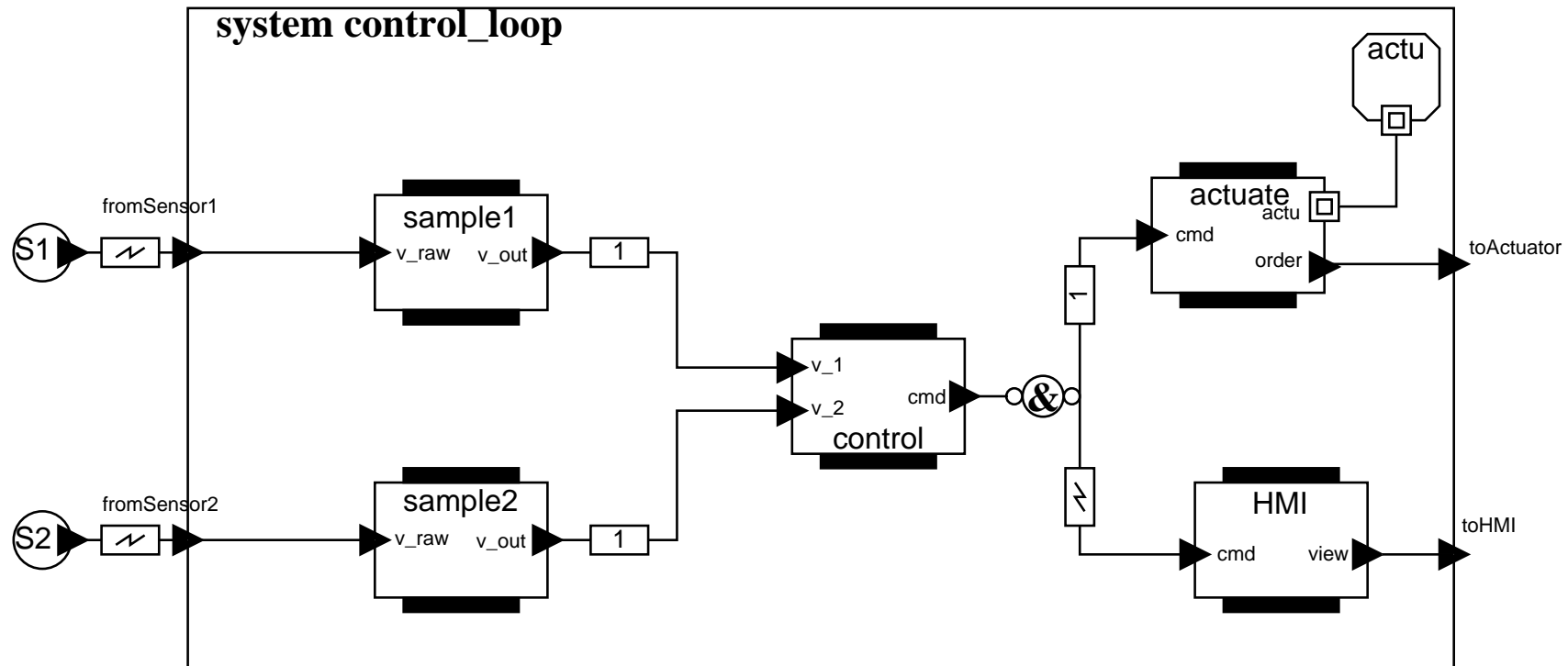
A possible CLARA description



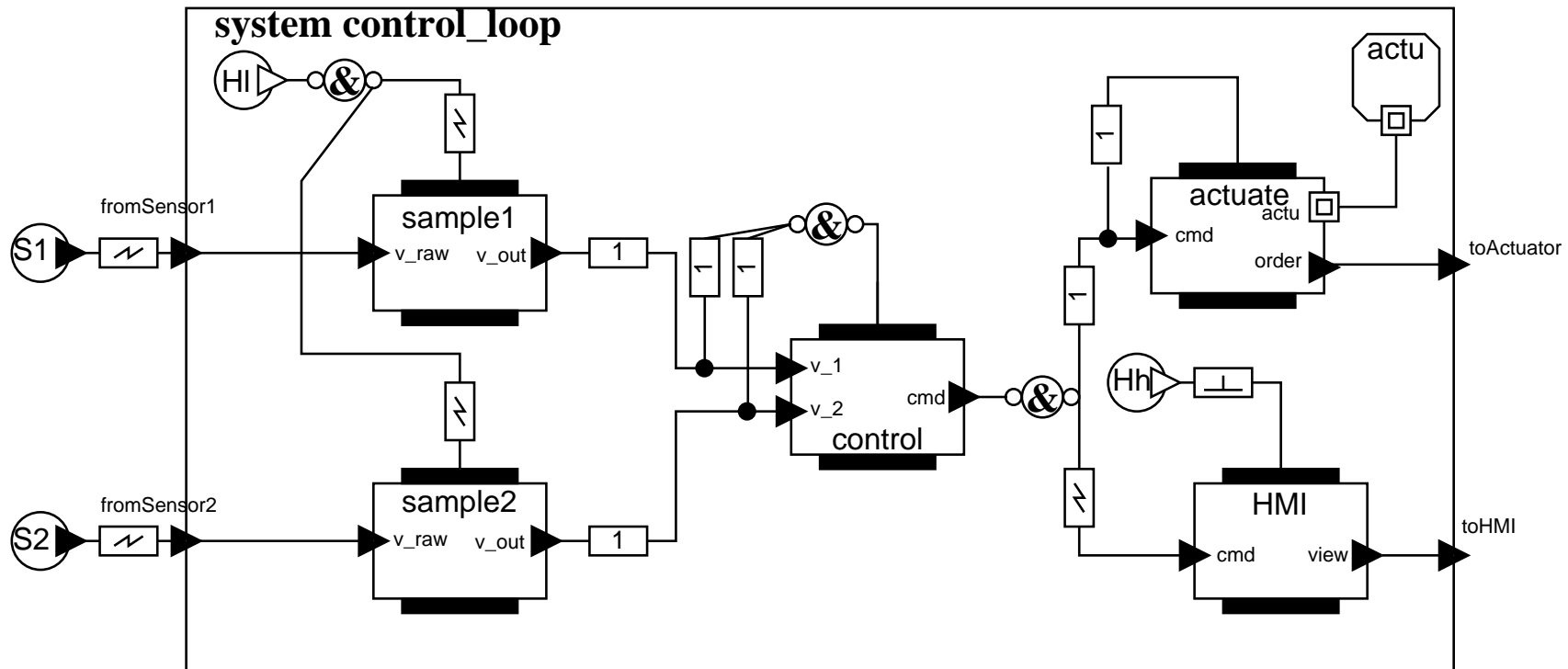
A possible CLARA description



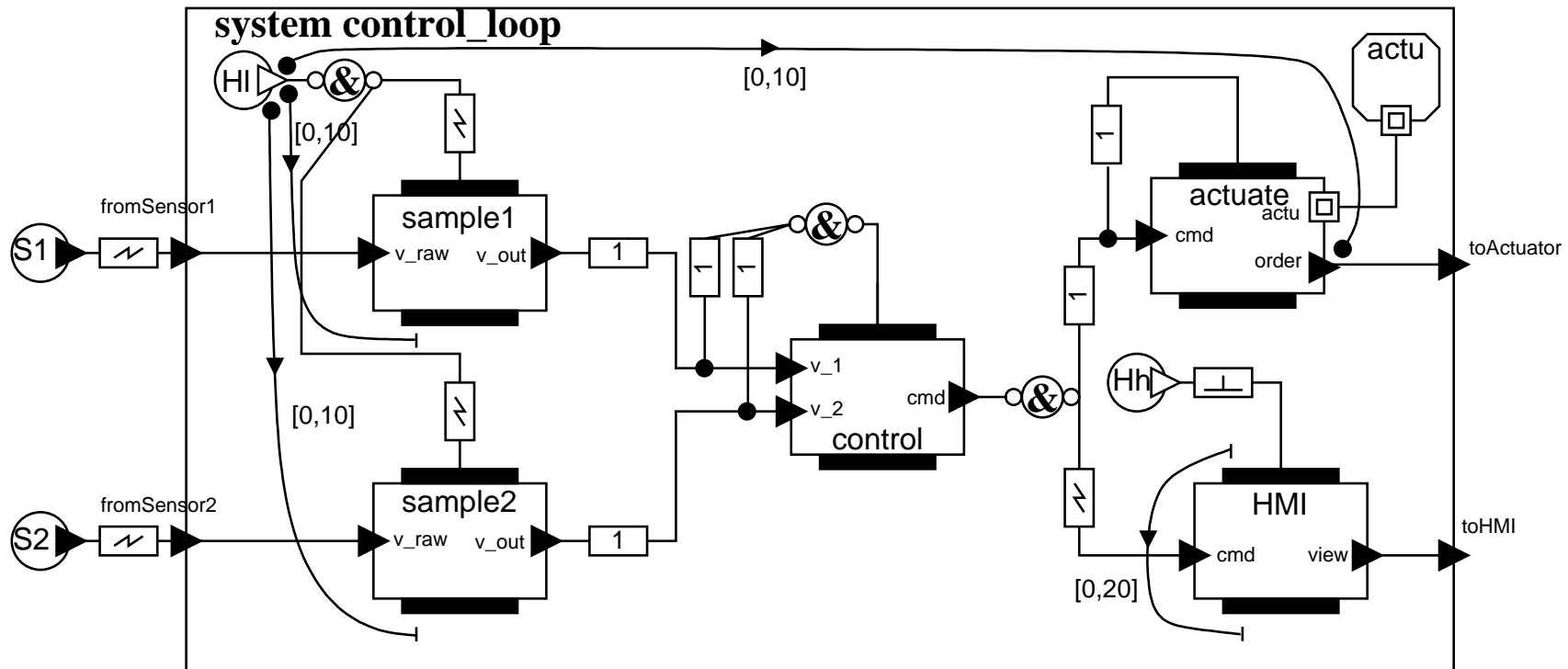
A possible CLARA description



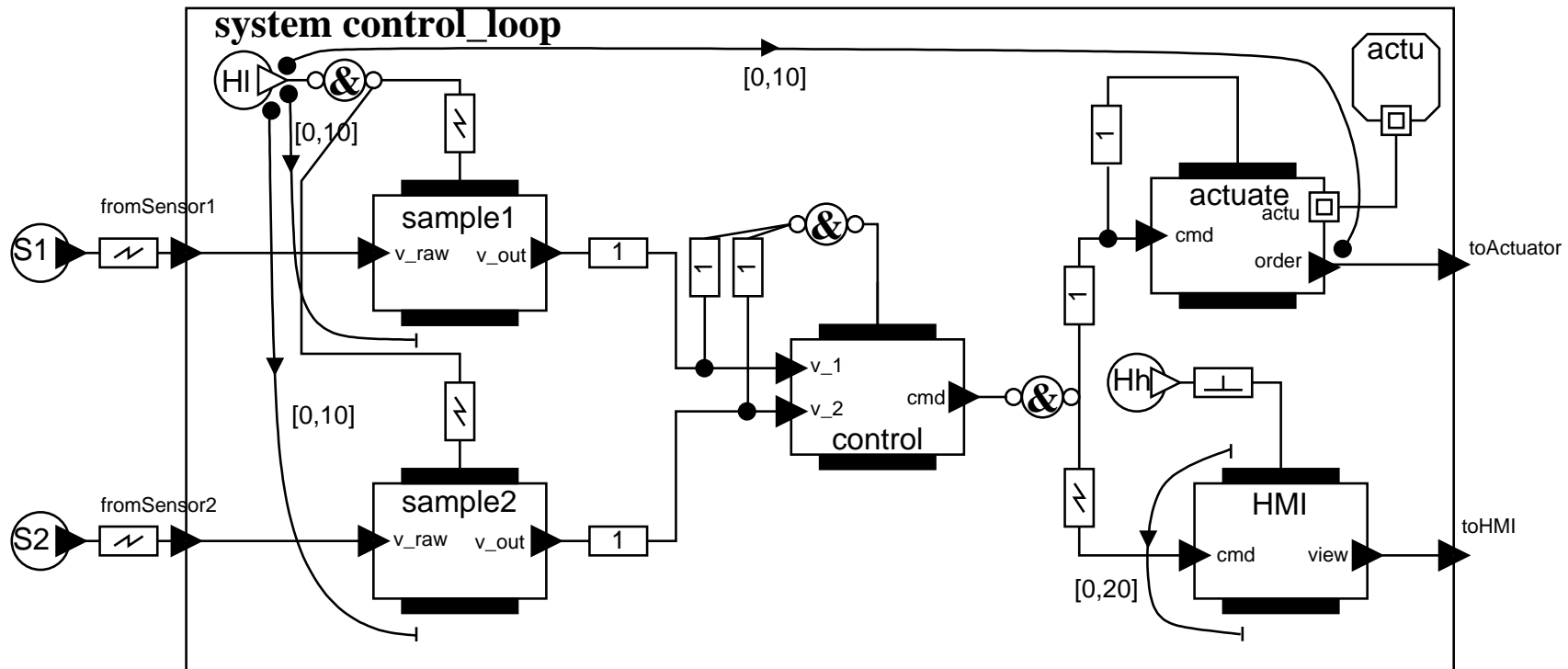
A possible CLARA description



A possible CLARA description



A possible CLARA description



Comprehensive description of the system control flows

└ A possible CLARA description (2)

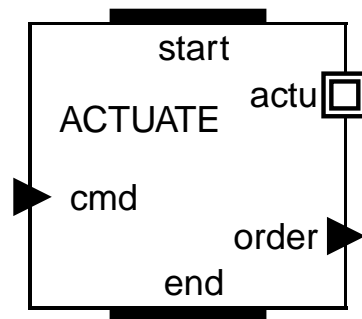
What about the behaviours of activities?

- ▶ each (atomic) activity is associated to a **finite sequence of actions**
- ▶ actions are either **port-related** (read, write, etc.) or call of a **user-defined function** (piece of code)
- ▶ each function has a **time budget** (closed interval)

└ A possible CLARA description (2)

What about the behaviours of activities?

- ▶ each (atomic) activity is associated to a **finite sequence of actions**
- ▶ actions are either **port-related** (read, write, etc.) or call of a **user-defined function** (piece of code)
- ▶ each function has a **time budget** (closed interval)



```
name ACTUATE;  
sequence {  
  read(cmd);  
  access(actu.get);  
  call(translate_command,0.5,1);  
  write(order);  
  access(actu.release);  
}
```

Validation of the FA

What needs to be validated?

- ▶ quality of the design: conformance to a style, complexity measures, etc.
- ▶ **correctness of the design**: liveness and safety properties
- ▶ **feasability of the design**: e.g. coherency between timing constraints and budgets

Validation of the FA

What needs to be validated?

- ▶ quality of the design: conformance to a style, complexity measures, etc.
- ▶ **correctness of the design**: liveness and safety properties
- ▶ **feasability of the design**: e.g. coherency between timing constraints and budgets

Remark:

- ▶ Safety-critical system \Rightarrow Formal analysis
- ▶ Formal analysis \Rightarrow Formal semantics

CLARA architecture \rightsquigarrow (Time) Petri Net model

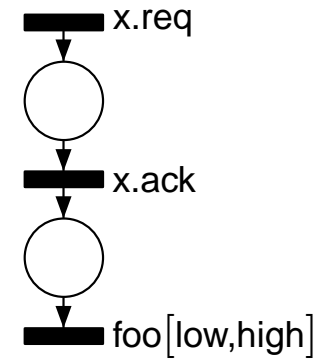
└ From CLARA arch. to PN model (1)

Step 1: every entity is associated to a PN pattern

From CLARA arch. to PN model (1)

Step 1: every entity is associated to a PN pattern

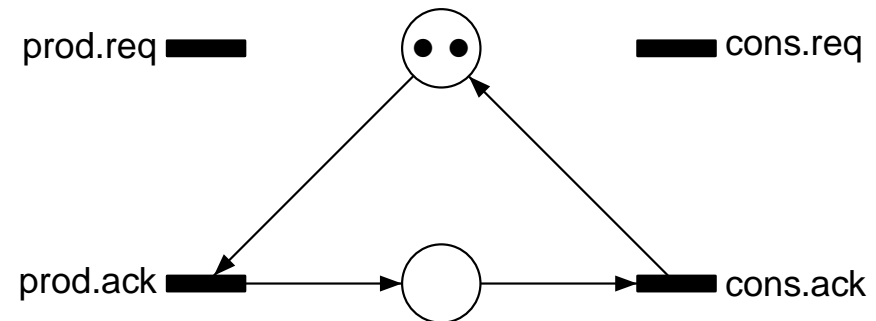
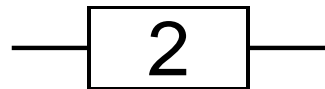
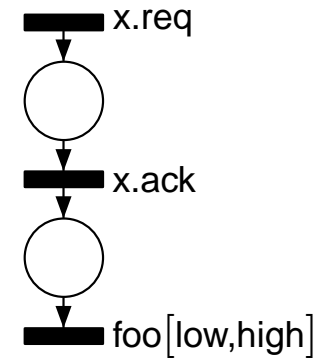
...
read(x);
call(foo, low, high);
...



From CLARA arch. to PN model (1)

Step 1: every entity is associated to a PN pattern

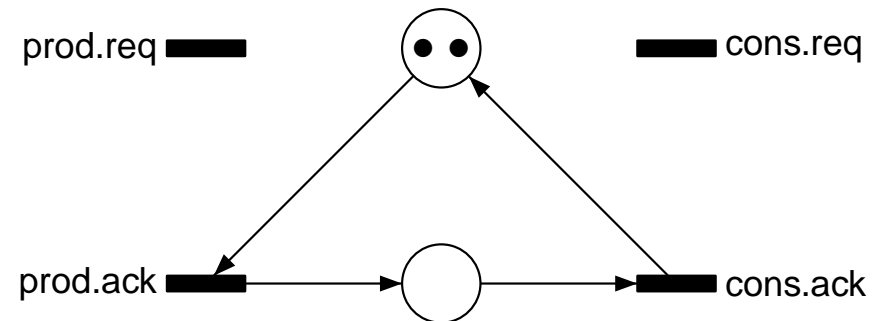
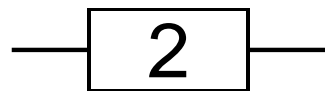
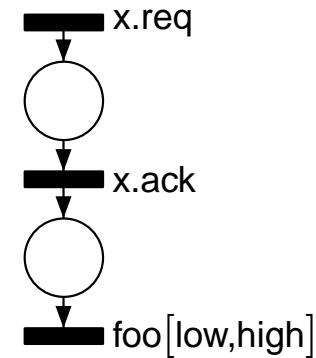
...
read(x);
call(foo, low, high);
...



From CLARA arch. to PN model (1)

Step 1: every entity is associated to a PN pattern

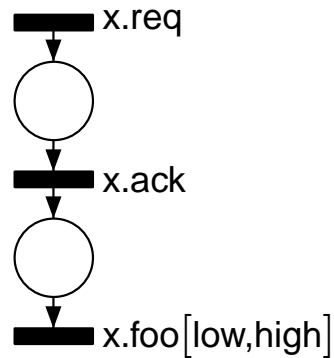
...
read(x);
call(foo, low, high);
...



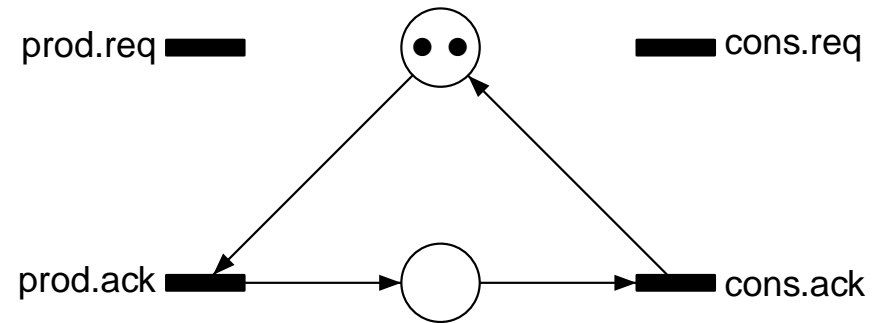
etc.

From CLARA arch. to PN model (2)

Step 2: transitions merging

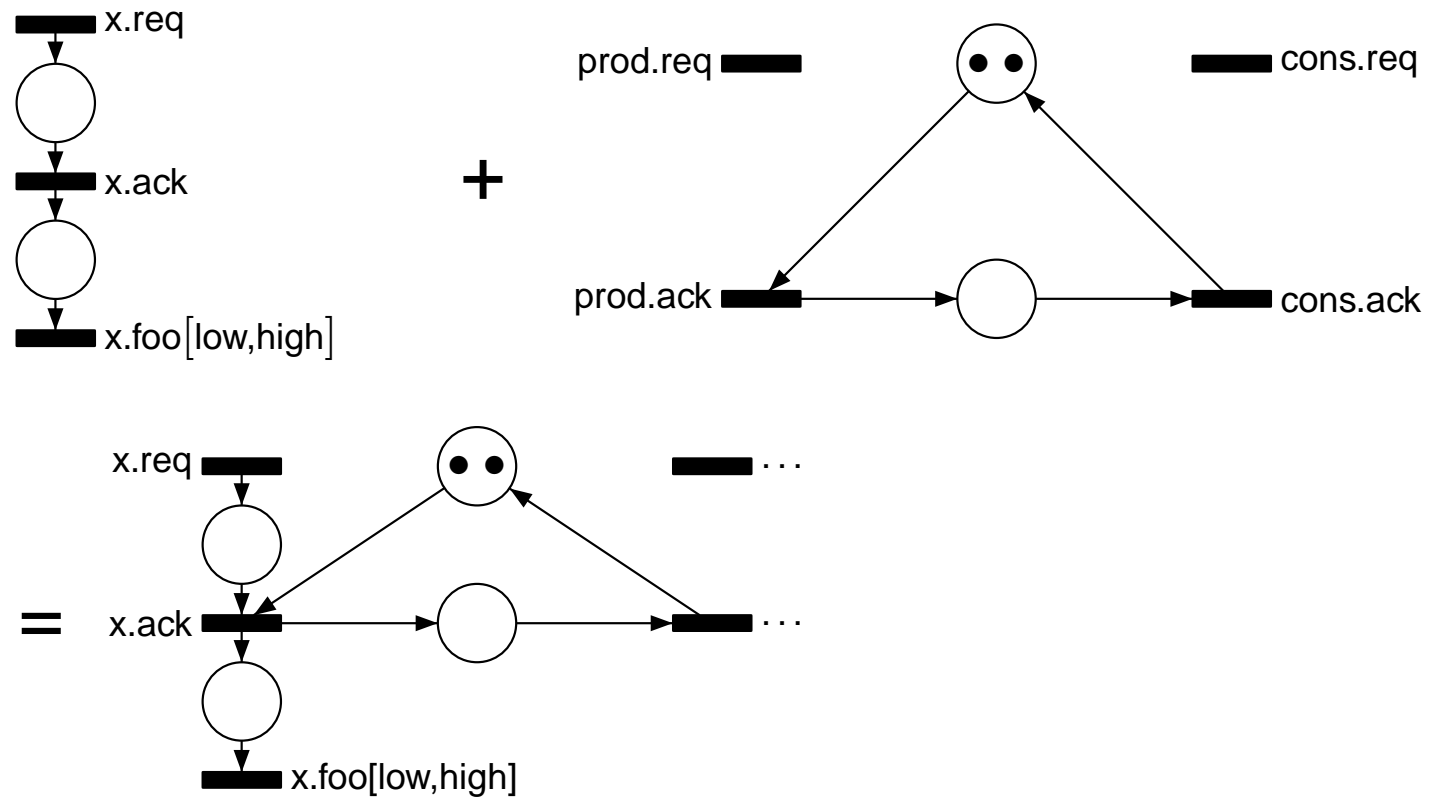


+



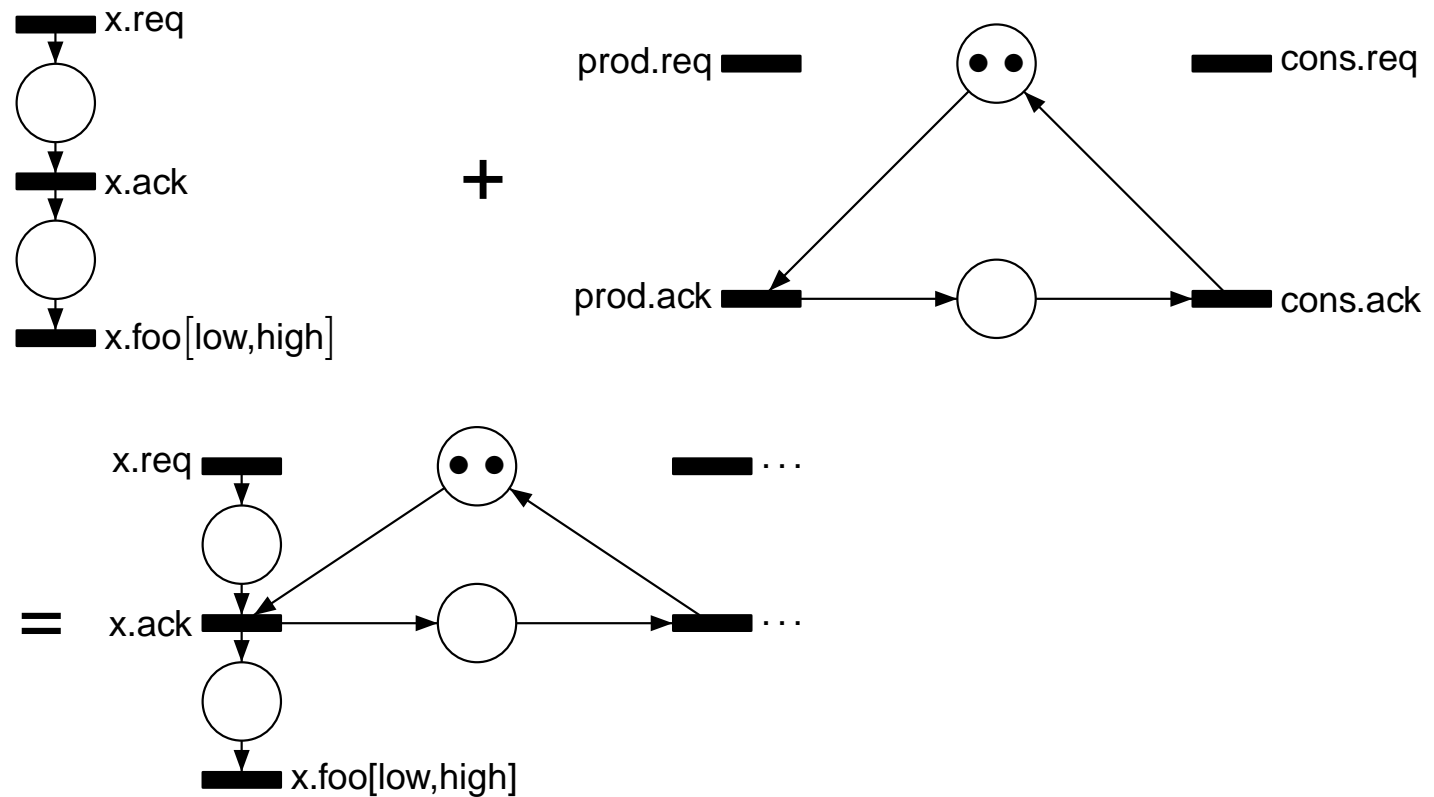
From CLARA arch. to PN model (2)

Step 2: transitions merging



From CLARA arch. to PN model (2)

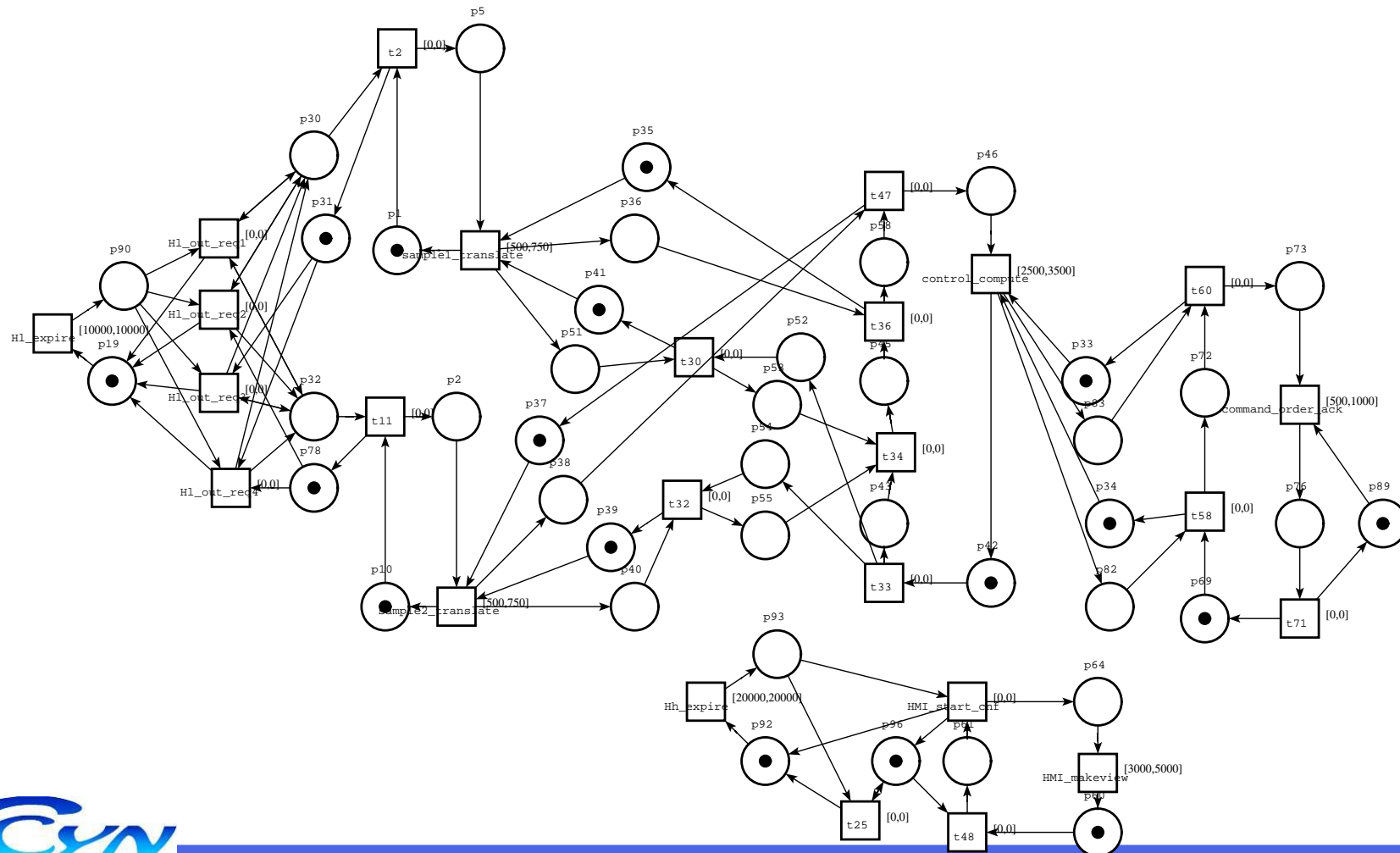
Step 2: transitions merging



Step 3: reduction (discard useless places and transitions)

Back to the example

After reduction, we obtain this PN model (42 places, 25 transitions):



└ Correctness: PN model analysis

1. Marking graph computation and analysis?

- ▶ PN model **simulates** any possible (correct) implementation: **analysis limited to safety properties**
- ▶ tools: Roméo or TINA (computation), CADP or MEC (analysis)

└ Correctness: PN model analysis

1. Marking graph computation and analysis?

- ▶ PN model **simulates** any possible (correct) implementation: **analysis limited to safety properties**
- ▶ tools: Roméo or TINA (computation), CADP or MEC (analysis)

2. Structural analysis ?

- ▶ used to detect repetitive behaviours (t-semi-flows generating sets analysis): are they conform to what we expect?
- ▶ tool: TINA

└ Feasibility: coherency of timing info.

What we **cannot do**: decide schedulability (wrong design level)

What we **can do**: prove that a schedulable implementation may exist (**necessary condition**)

- ▶ **Idea**: for each constrained chain, compare a lower bound of its execution time to the upper bound of the constraint
- ▶ **Technique**:
 1. build precedence graph of the elementary functions
 2. path analysis of this graph to obtain the lower bound

Summary

CLARA

- ▶ description of **FA** rather than SA (\neq MetaH/AADL or Cotre)
- ▶ **accurate and comprehensive** description of control flows
- ▶ formal operational semantics

Validation

- ▶ **Correctness**: through classical Petri Nets theory
- ▶ **Feasibility**: necessary condition for coherency between constraints and budgets
- ▶ Both kind of analysis are **limited by the design level targeted**

└ On-going and future works

On-going

- ▶ graphical editor and related tools (**GME-based**)
- ▶ automatic production of valid operational architecture (constraint programming + sched. analysis)

Future

- ▶ analysis of operational architectures (SETPN)
- ▶ carry on the work up to code synthesis facilities