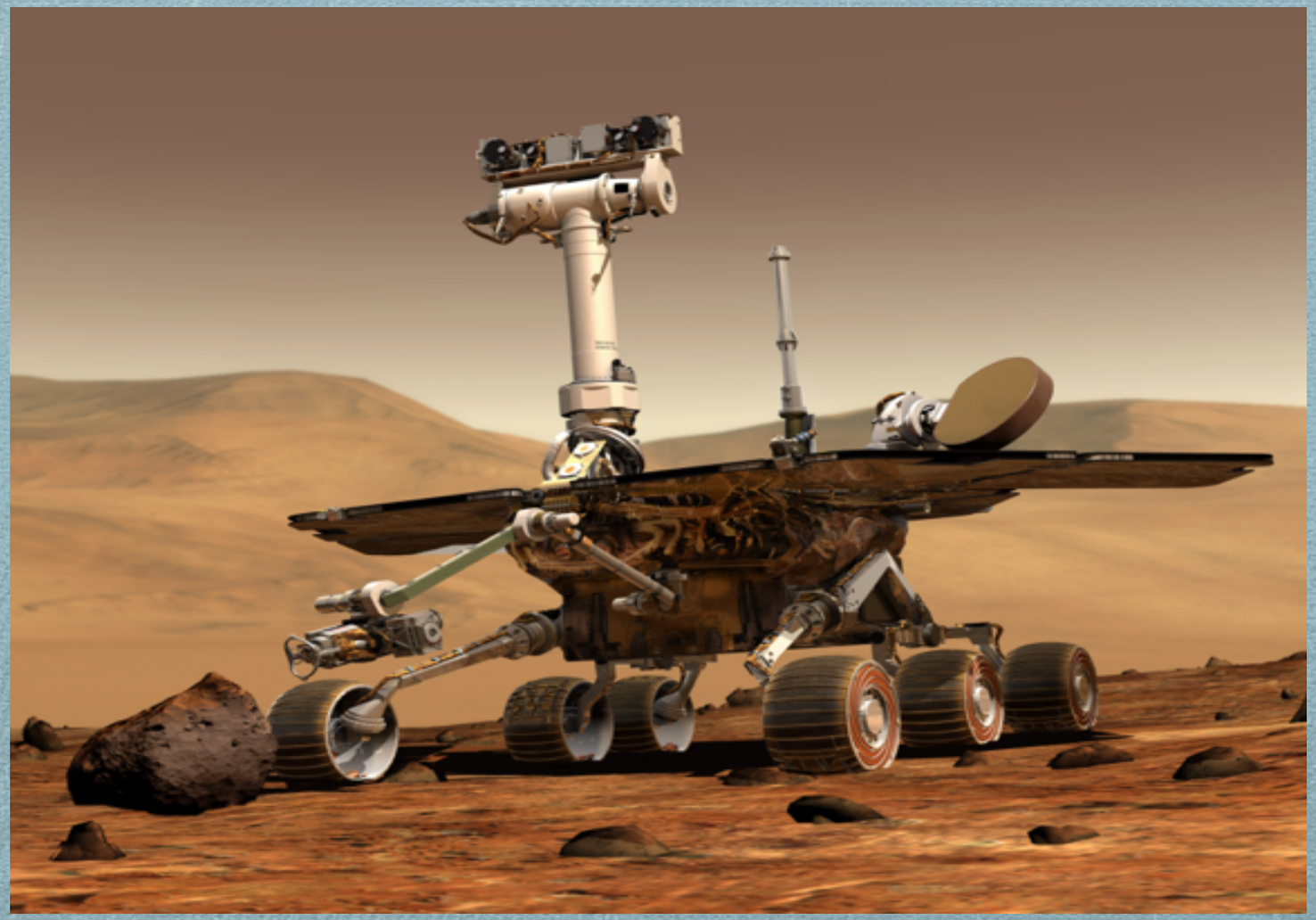
Temporal/Timed Formal Verification of Autonomous Robots

Mohammed Foughali -LAAS-CNRS

Content

I. Problem Definition

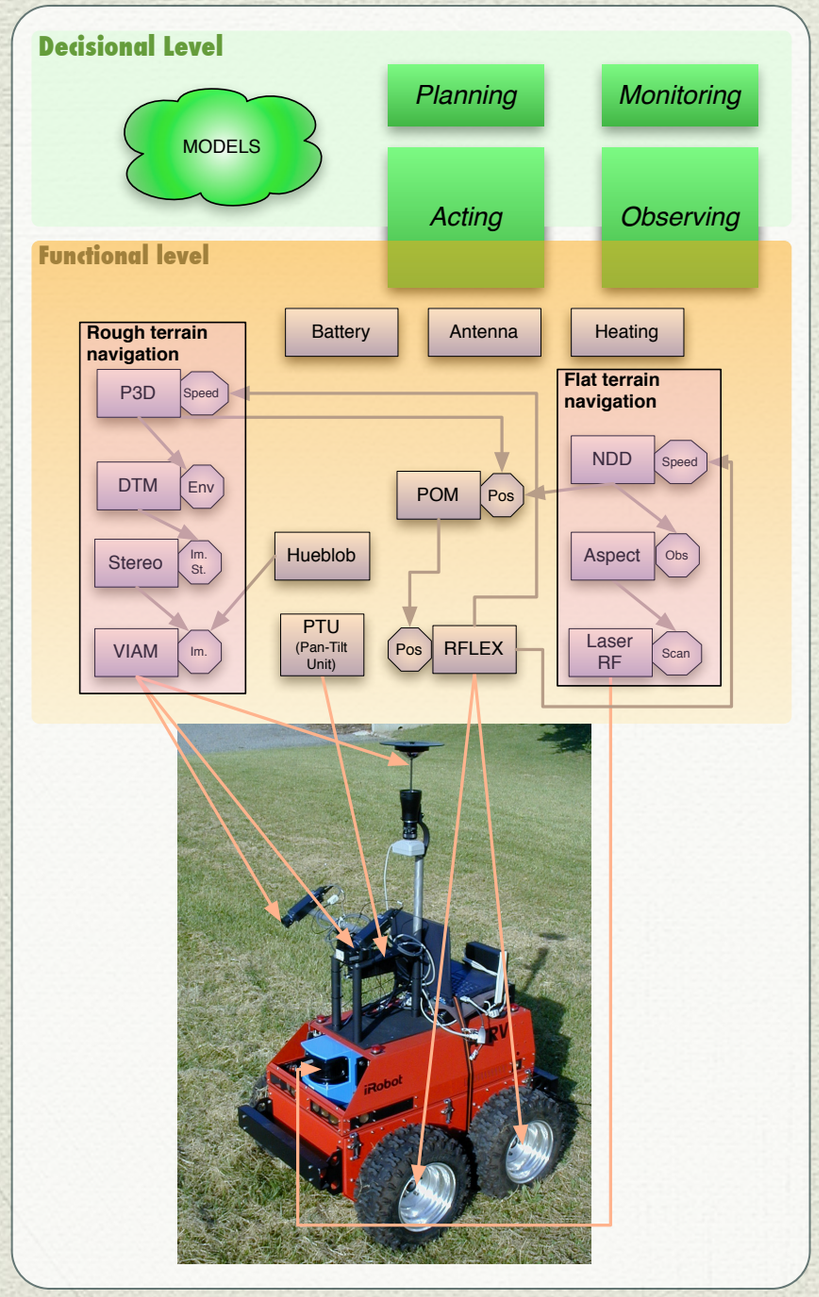


Autonomous robots -> high level of complexity

A growing need of formal guarantees on the systems' reliability as the robots are more and more involved in human environments and/or costly missions

I. Problem Definition

- Autonomous system software levels:
- Decisional layer
  - ➢Deals with high-level missions such as planning, acting, etc.
  - ➢Often formal
- Functional layer
  - ➢Interacts directly with sensors and actuators
  - ➢Deployed via non formal frameworks (GenoM, ROS, etc.)
  - ➢Little has been done to formally verify its components



I. Problem Definition

- So far, roboticists rely heavily on simulation and tests
  - X Possibility to miss faulty execution paths => catastrophic damage to the robot, to the environment and/or, more dramatically, to humans.
- Formal verification offers mathematical guarantees
  - X Highly complex systems imply a costly investment in their formal modeling but also an explosion of the reachable state space
  - X Constrained formal frameworks (if used directly for specification)

I. Problem Definition

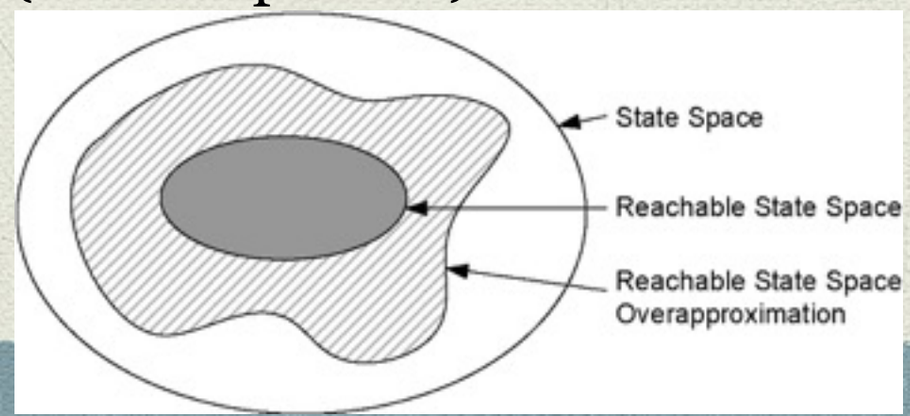
Examples of such limits on related works:

➡ Model Checking

- Espiau et al. (1995)
  - ❖ Orccad -> ESTEREL -> Mauto (untimed verification)
  - ❖ Orccad -> Timed Argos -> Kronos (timed verification)
    - X Properties verified on very simple examples under the threat of explosion
    - X The time-consuming and error-prone formal modeling step needs to be re-done for every new example

➡ Compositional Verification

- GenoM/BIP experiments, Ingrand et al. (2005-2012)
  - ❖ Over-approximation of the reachable state space (avoid explosion)
  - ❖ GenoM -> BIP -> D-Finder
    - X Time constraints forgotten
    - X Can't decide on properties evaluated as false

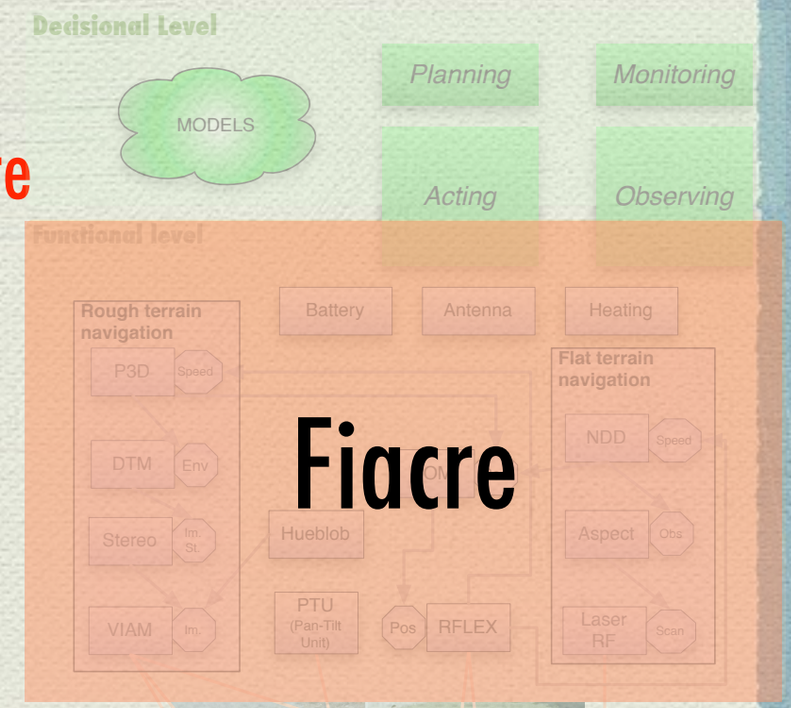


II. Our Approach

LAAS/RIS

- Functional level : GenoM
- Modules
  - Services (control flow)
  - Ports (data flow)

Model-Driven Software Engineering



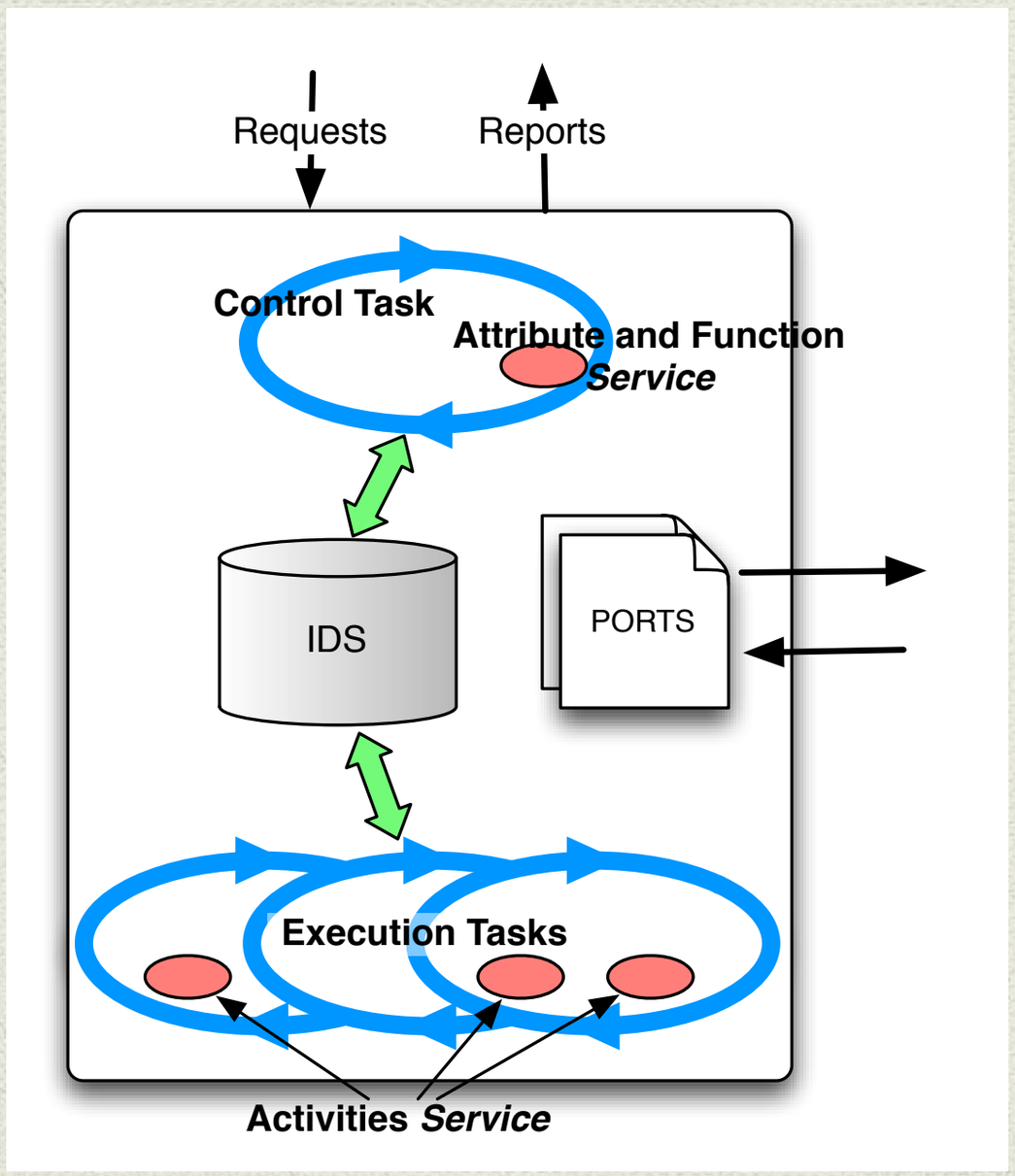
LAAS/VerTICS

- Fiacre/TINA framework for time-constrained distributed/concurrent systems

Formal Methods

II. Our Approach

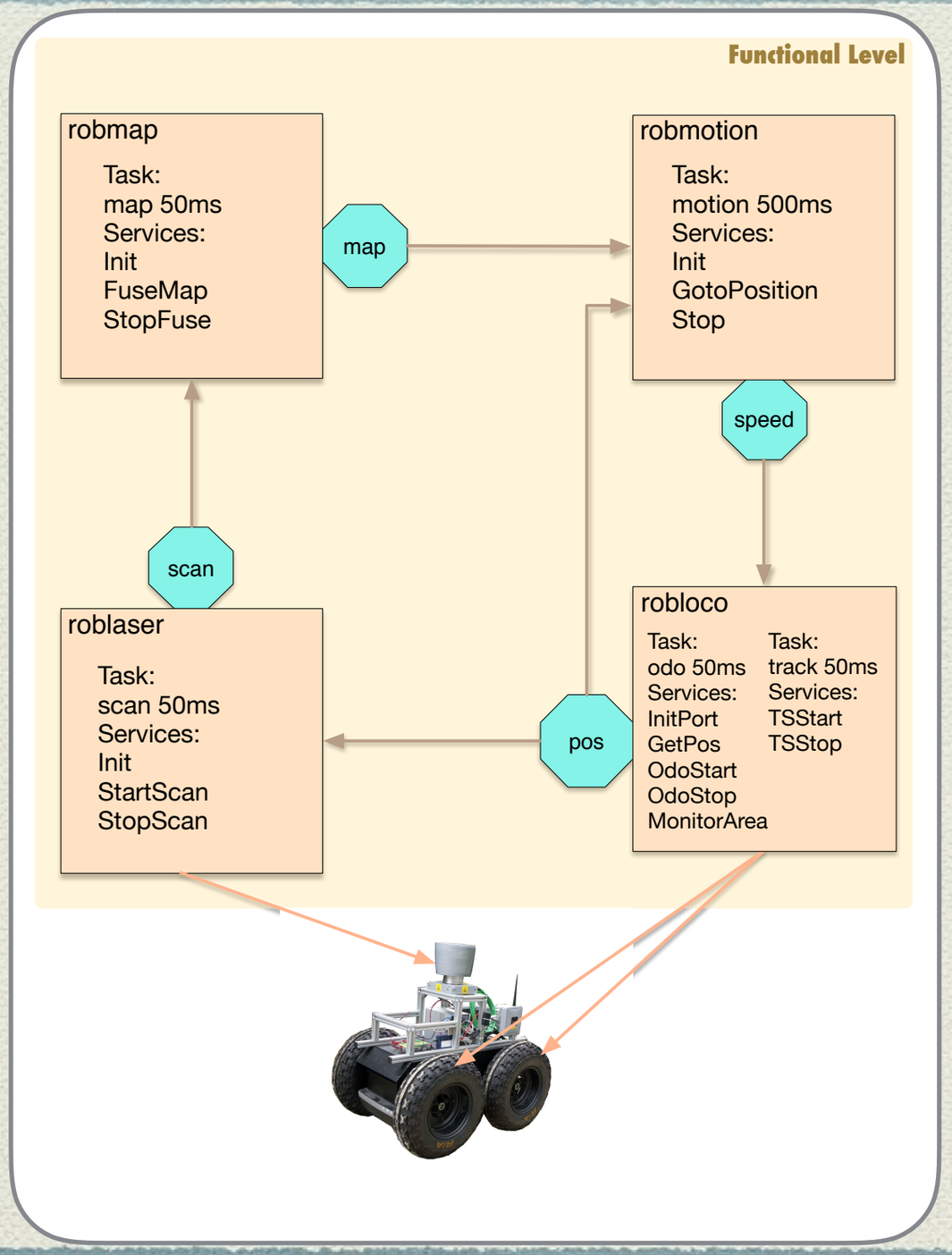
- GenoM
  - Services (control flow)
  - Ports (data flow)
  - Activities (automata)
  - Control task
  - Execution tasks



II. Our Approach

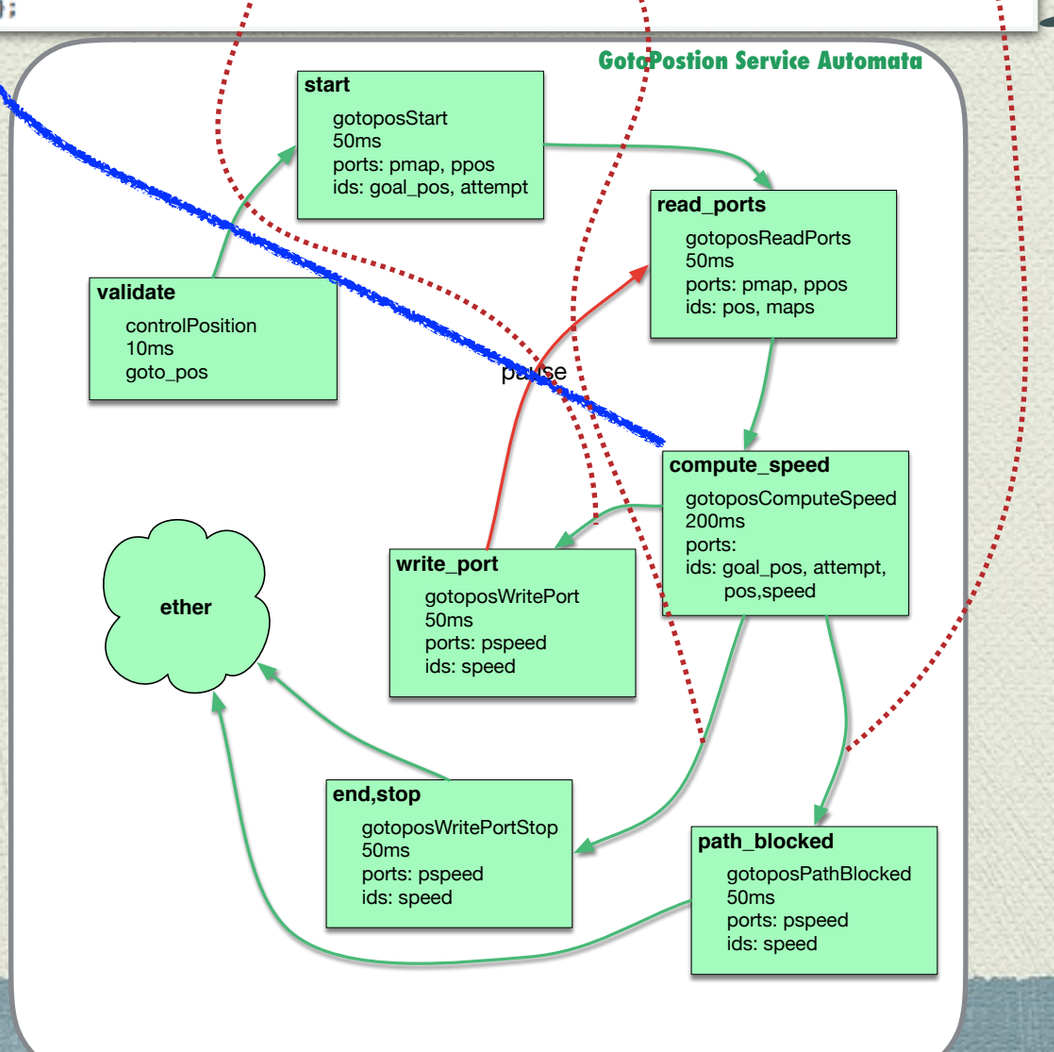
Example

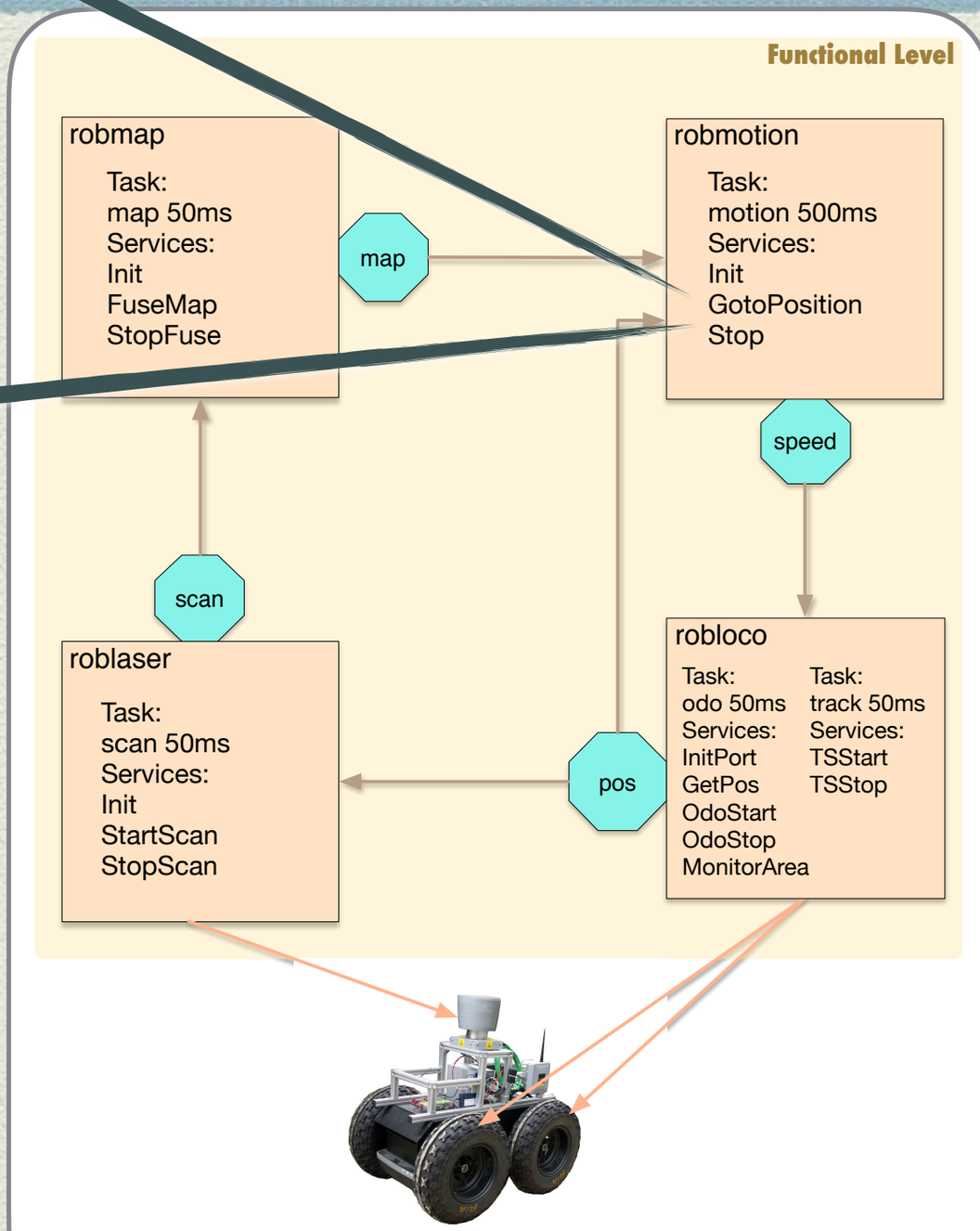
- 4 modules for robot navigation
- 4 ports
- 4 control task and 5 execution tasks
- >16 services



II. Our Approach

```
activity GotoPosition (in robloco::position goto_position =  :"Goto position")
{
 doc                  "Move to the given position";
 validate             controlPosition (in goto_position) wcet 10 ms;
 codel <start>        gotoposStart(in goto_position, port in ppos, port in pmap,
                                   ids out goal_pos, ids out attempts)
                        yield read_ports, ether wcet 50 ms;
 codel <read_ports>   gotoposReadPorts(ids out pos, ids out explored_map,
                                       port in ppos, port in pmap)
                        yield compute_speed wcet 50 ms;
 codel <compute_speed> gotoposComputeSpeed(ids in goal_pos, ids in pos,
                                           ids in explored_map, ids in verbose,
                                           ids out speed, ids inout attempts)
                        yield write_port, end, path_blocked wcet 200 ms;
 codel <write_port>   gotoposWritePort(ids in speed, port out pspeed)
                        yield pause::read_ports wcet 50 ms;
 codel <end,stop>     gotoposWritePortStop(ids out speed, port out pspeed)
                        yield ether wcet 50 ms;
 codel <path_blocked> gotoposPathBlocked(ids out speed, port out pspeed)
                        yield ether wcet 50 ms;
 interrupts           GotoPosition;
 task                 plan;
 throw                INVALID_POSITION, INVALID_MAP, INVALID_GOAL, PATH_BLOCKED;
};
```





II. Our Approach

- Template mechanism: GenoM provides a template-based generator to translate a GenoM specification into other representations

- Modules can be generated for different middleware (ROS-Com, PocoLibs, etc.)

➡ Program templates to bridge GenoM with V&V Tools and Frameworks

II. Our Approach

- Fiacre (Format Intermédiaire pour les Architectures de Composants Répartis Embarqués)
  - **Timed discrete-event systems coding based on Automata and Time Petri Nets**
  - **Communication and synchronization through ports and shared variables**
  - **Possibility to formulate LTL properties**
  - **Patterns: possibility to express timed properties**

II. Our Approach

- **Example of communicating Fiacre processes: The Fischer protocol**

```
/* Processes */

process Proc (pid : id, &lock : lock) is
  states WaitLock, WaitLock2, SetLock,
TestLock, CriticalSection

  from WaitLock
    on (lock = 0);
    to WaitLock2

  from WaitLock2
    wait [0, 2];
    lock := pid;
    to SetLock

  from SetLock
    wait ]2, ...[;
    to TestLock

  from TestLock
      if  lock = pid then
            to CriticalSection
      else
            to WaitLock
      end

  from CriticalSection
      lock := 0;
      to WaitLock

/* Main component */
component Main is
  var lock : lock := 0
  par
     Proc (1, &lock)
  || Proc (2, &lock)
  end

* Entry point for verification */
Main

/* Mutual exclusion */
property mutex is ltl [] not ((Main/1/state CriticalSection) and (Main/2/state
CriticalSection))
```

II. Our Approach

- TINA (TIme Petri Net Analyzer)
- A toolbox for the editing and analysis of **Time Petri Nets and Time Transition Systems**

➢ LTL model-checking techniques

➢ Fiacre specification compiler

❖ FRAC (FiacRe to tinA Compiler)

✓ transform Fiacre specifications into Time Transition Systems (formally verifiable by TINA)

✓ Convert patterns into LTL properties

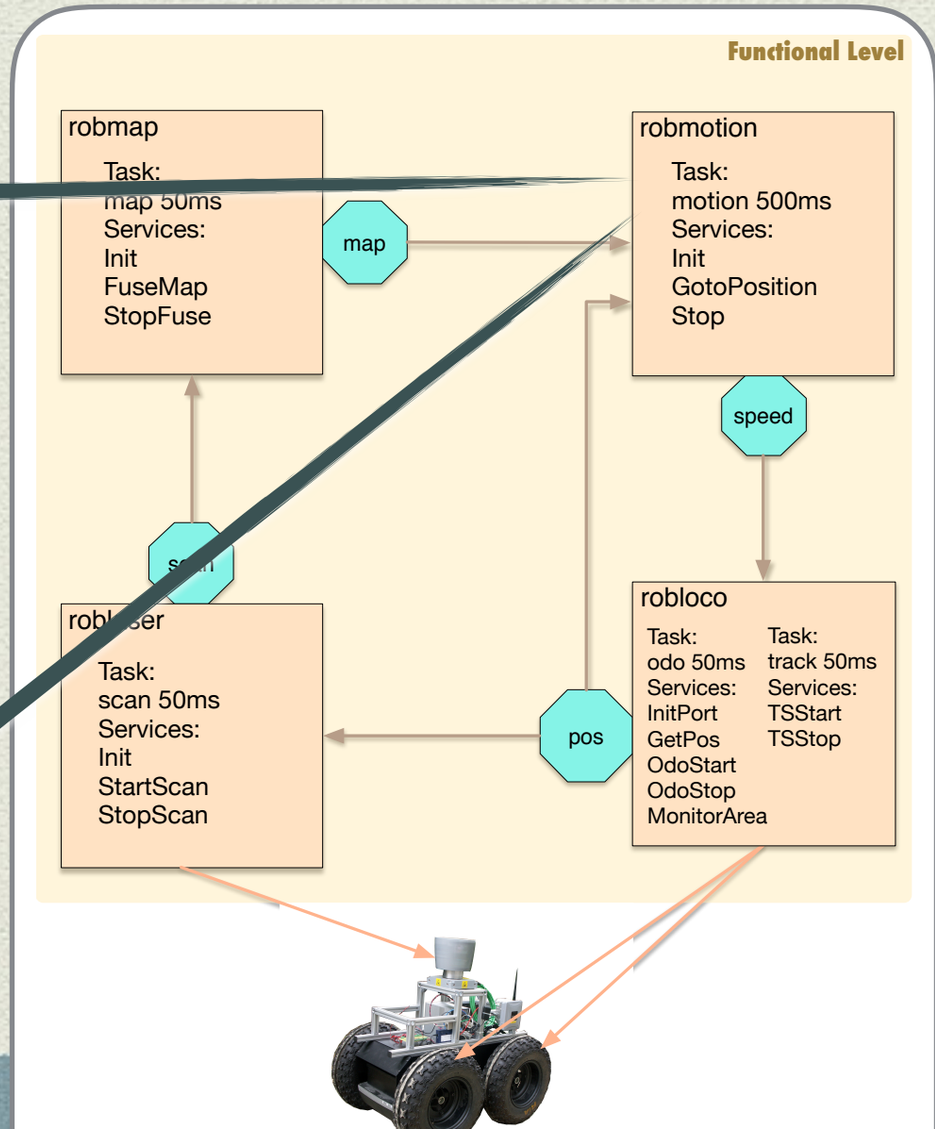
II. Our Approach

- A template that produces the Fiacre model out of any GenoM specification for the PocoLibs implementation

- example:

```
process timer (&tick: bool) is
states start
from start
wait [0.5,0.5];
tick := true;
to start
```

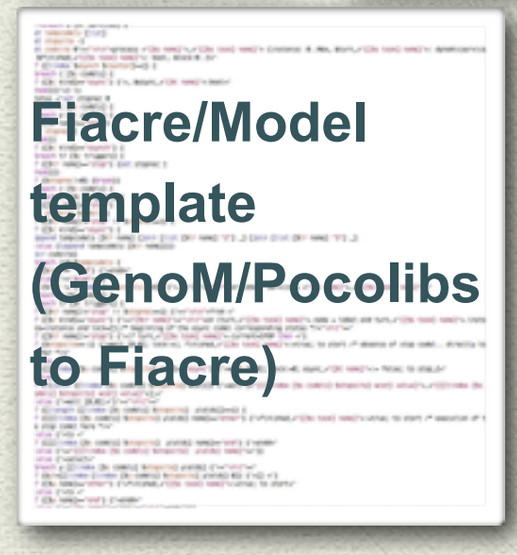
```
process Manager (&tick: bool, ...) is
states start, manage
from start
wait [0,0];
on tick;
tick := false;
if (...) /* no active activity */
then to start
else to manage end
from manage
wait [0,0];
... /* execute one active activity */
if (...) /* no more activities */
then to start
else to manage end
```



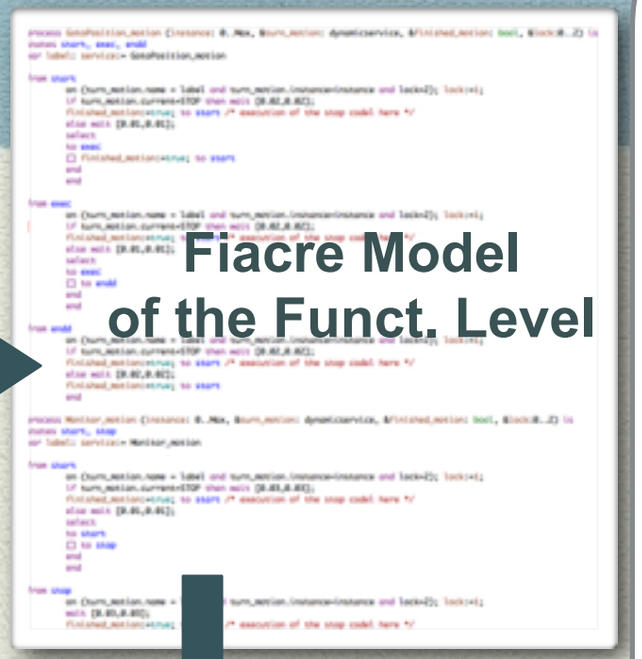
II. Our Approach

GenoM Models: .idl & .gen

Fiacre/Model template (GenoM/Pocolibs to Fiacre)



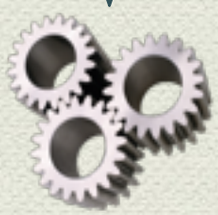
Fiacre Model of the Funct. Level



frac

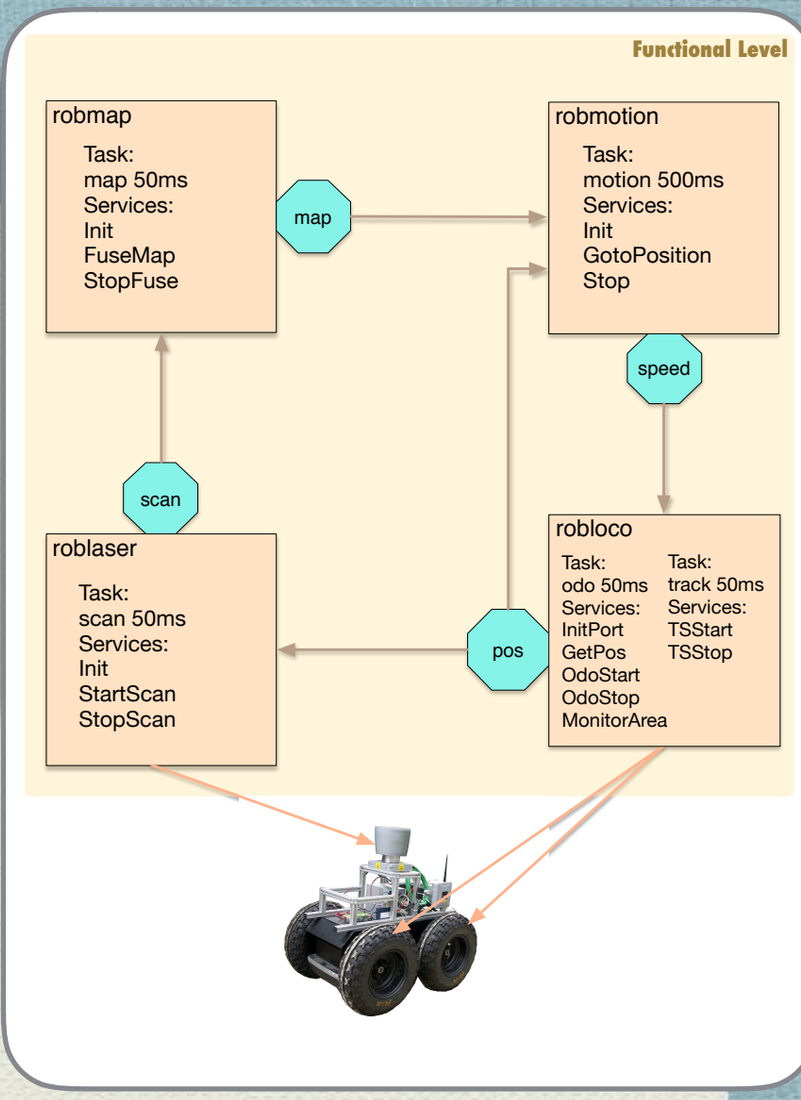
Real-time properties

Fix the model



TINA tools

Analysis

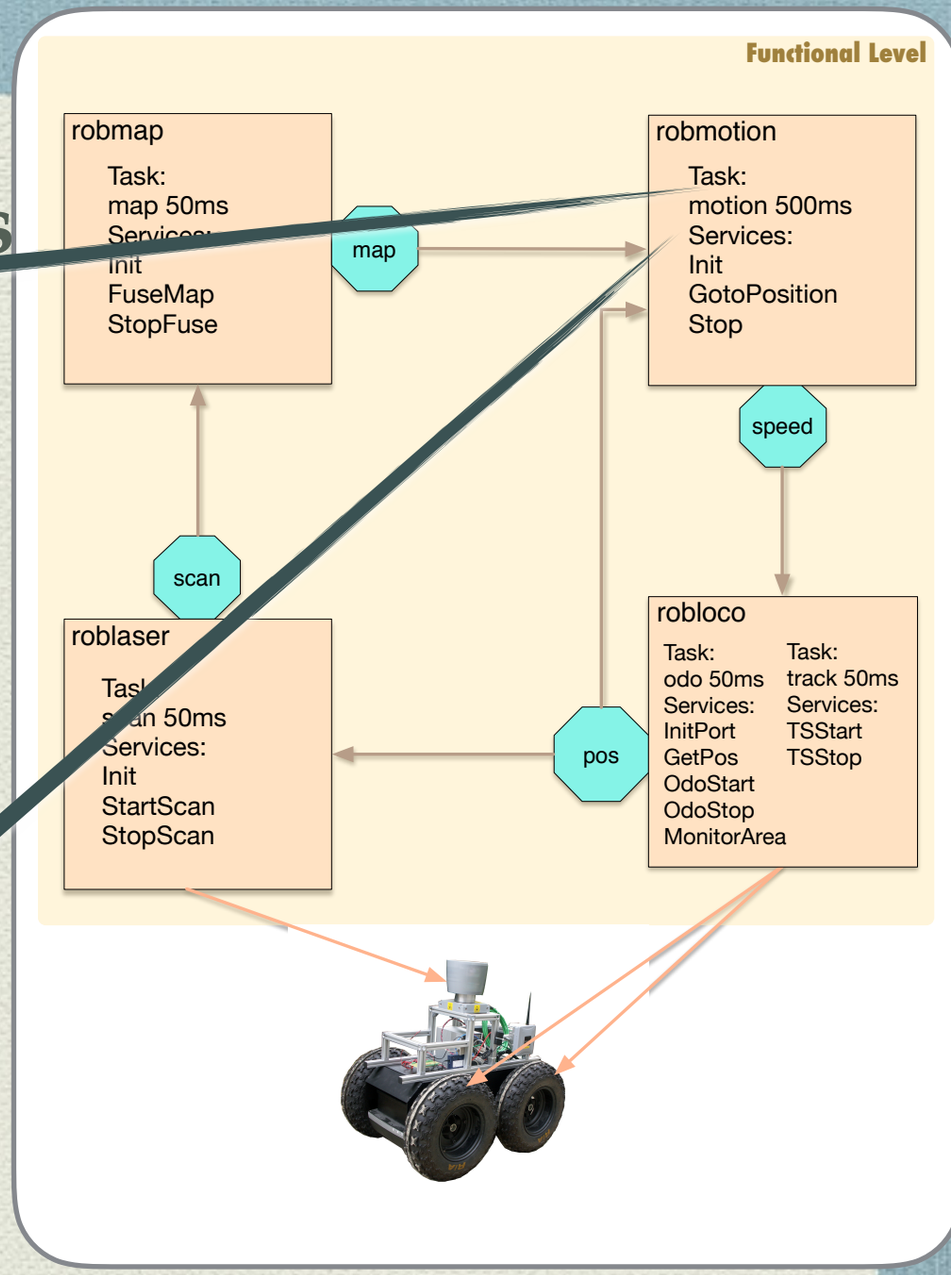


III. Verification Results

✓ Schedulability of execution tasks

```
process timer (&tick: bool) is
states start
from start
wait [0.5,0.5];
tick := true;
to start
```

```
process Manager (&tick: bool, ...) is
states start, manage
from start
wait [0,0];
on tick;
tick := false;
if (...) /* no active activity */
then to start
else to manage end
from manage
wait [0,0];
... /* execute one active activity */
if (...) /* no more activities */
then to start
else to manage end
```




```
property sched is always (navigation/robmap/manager/state manage) => not (navigation/robmap/manager/value tick)
```

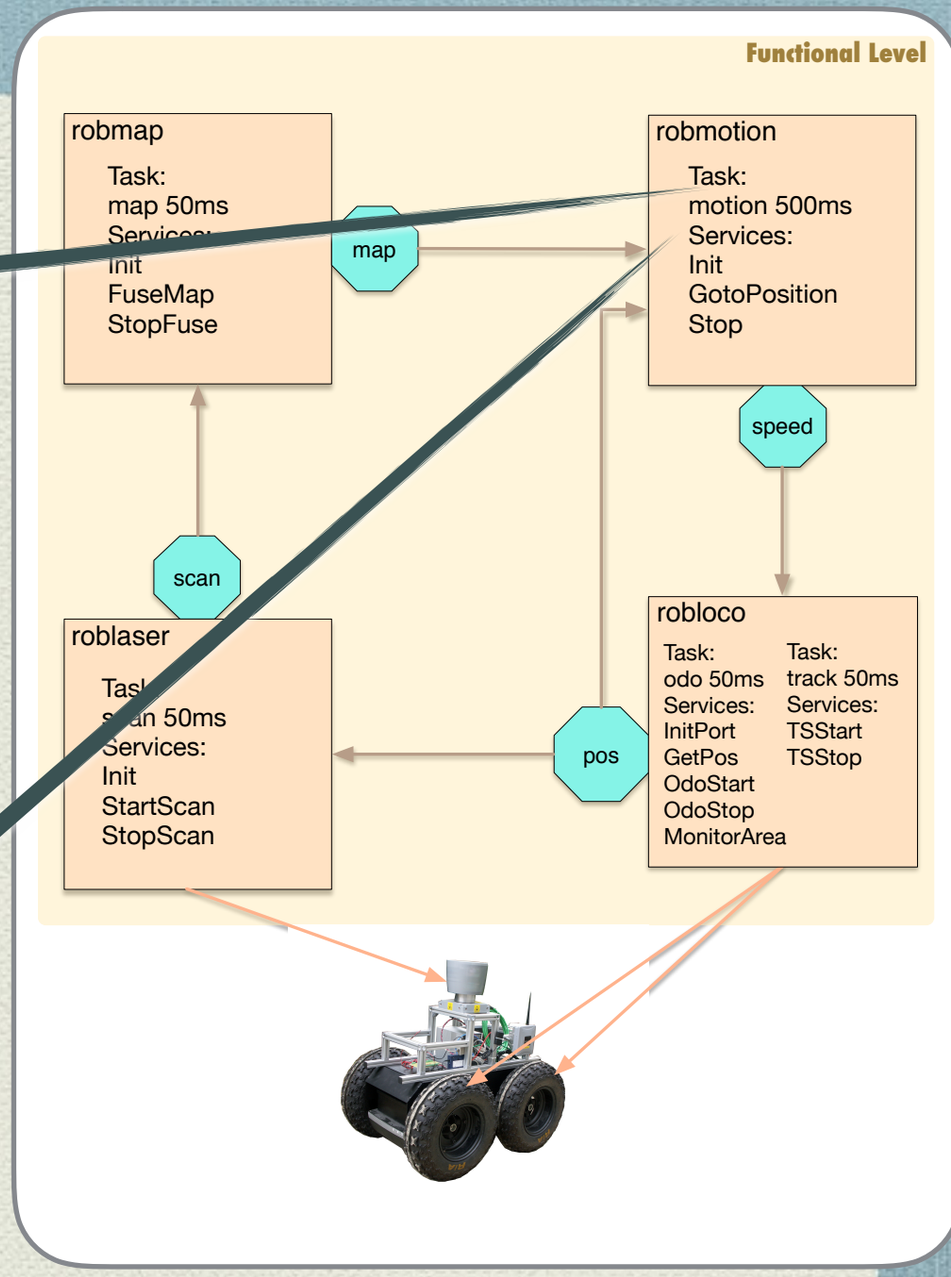
Verification with TINA: FALSE

III. Verification Results

✓ Progress of activities

```
process timer (&tick: bool) is
states start
from start
wait [0.5,0.5];
tick := true;
to start
```

```
process Manager (&tick: bool, ...) is
states start, manage
from start
wait [0,0];
on tick;
tick := false;
if (...) /* no active activity */
then to start
else to manage end
from manage
wait [0,0];
... /* execute one active activity */
if (...) /* no more activities */
then to start
else to manage end
```




```
property no_block is (navigation/robmap/manager/state manage) leadsto (navigation/robmap/manager/state start)
```

Verification with TINA: TRUE

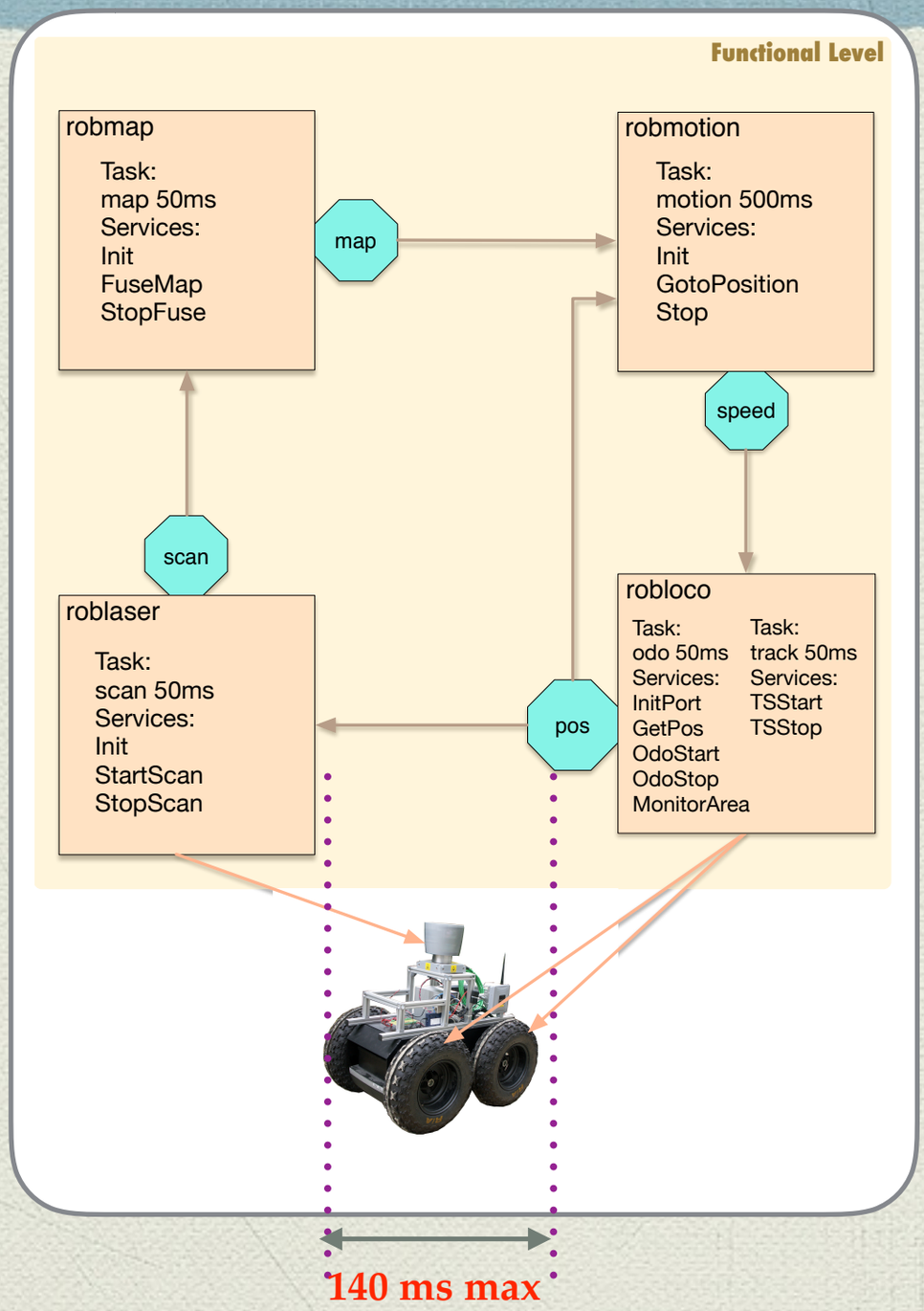
III. Verification Results

✓ **Position port update**

bounded in time

event A leads to event B within I

where I is an interval of integers

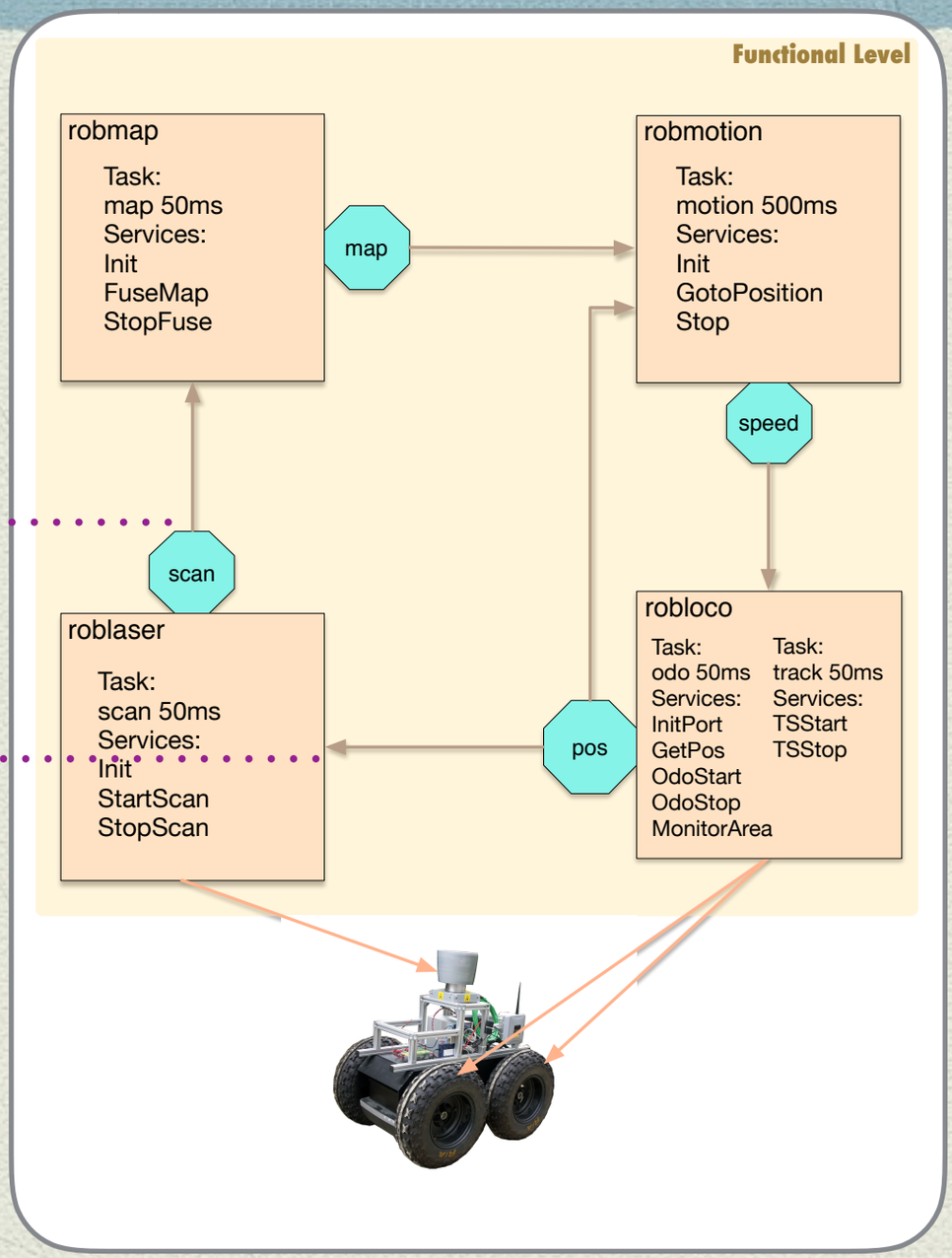


III. Verification Results

- ✓ **Position port update bounded in time**

event A leads to event B within I

where I is an interval of integers



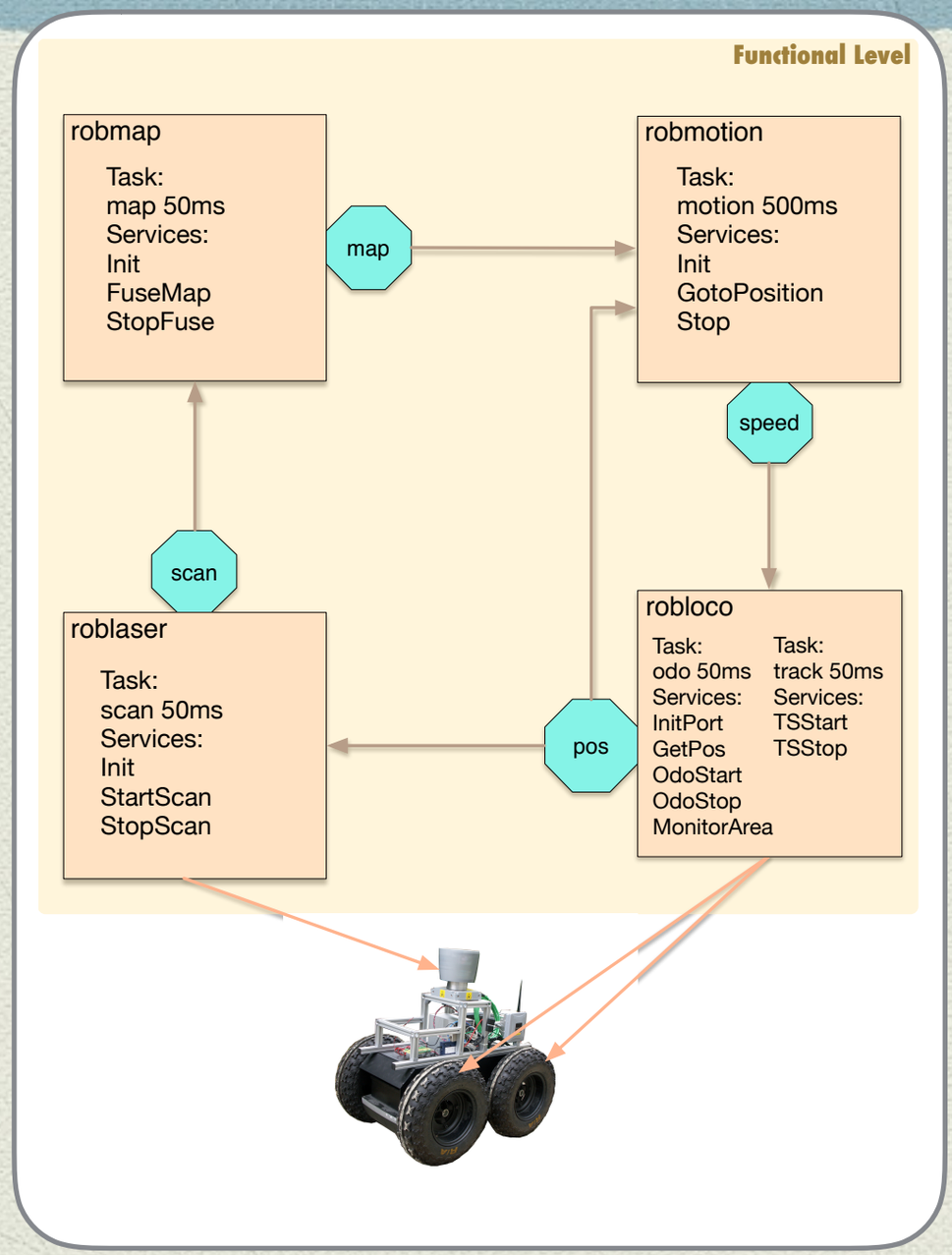
III. Verification Results

✓ **Position port update**

bounded in time

event A leads to event B within I

where I is an interval of integers



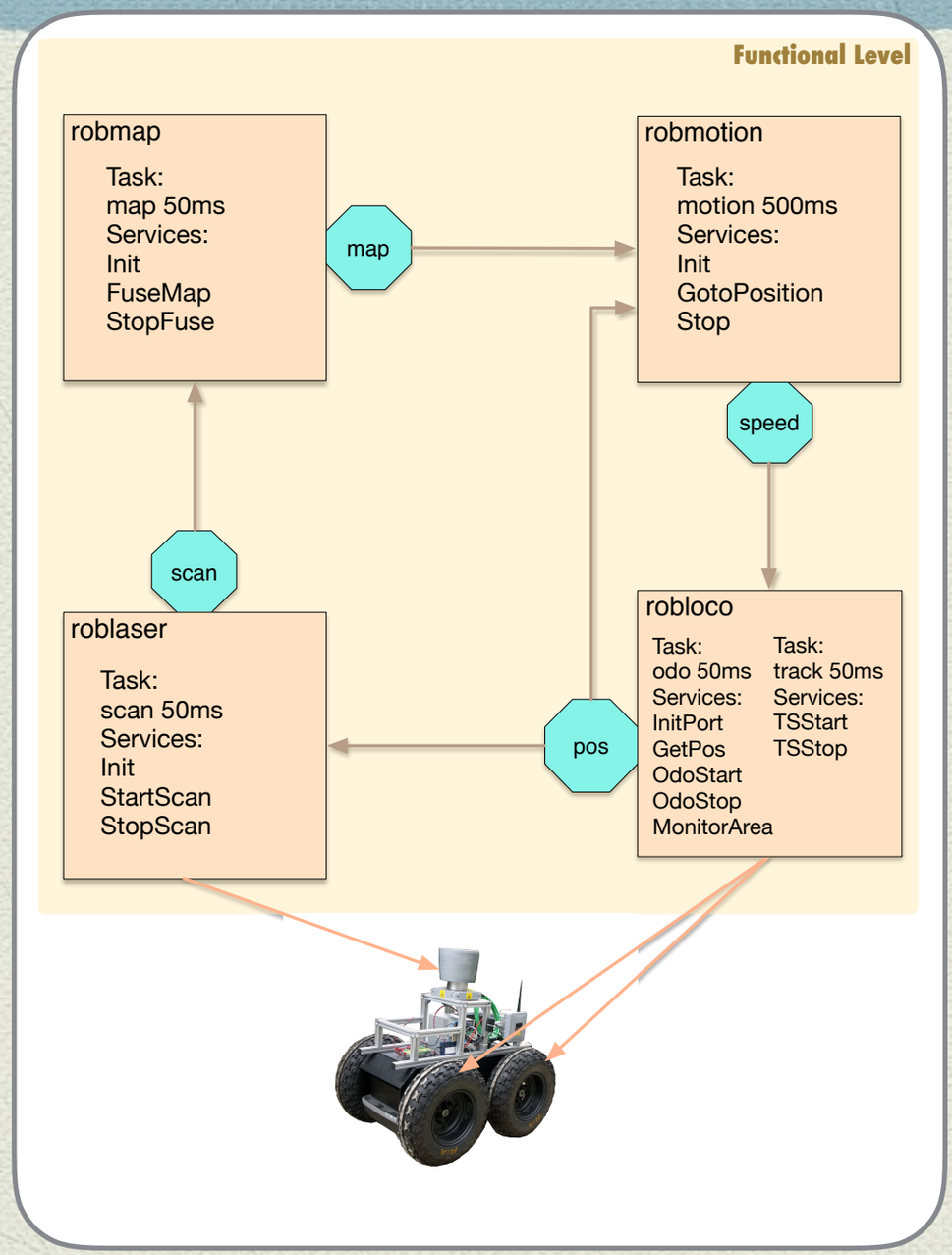
Final result → **1 second and 274 ms**

III. Verification Results

✓ Service termination bounded in time

If a Stop request is sent, the service TSStart will end within I

This leads to a null speed sent to the controller



Final result → **72 ms**

III. Verification Results

- ✓ **The roboticist analyzes the results and acts accordingly (if necessary)**

- ➡ **Example:**
  If the application is hard real-time, tune the periods so as all tasks become schedulable.

e.g doubling the period does it for the task ***track****..*

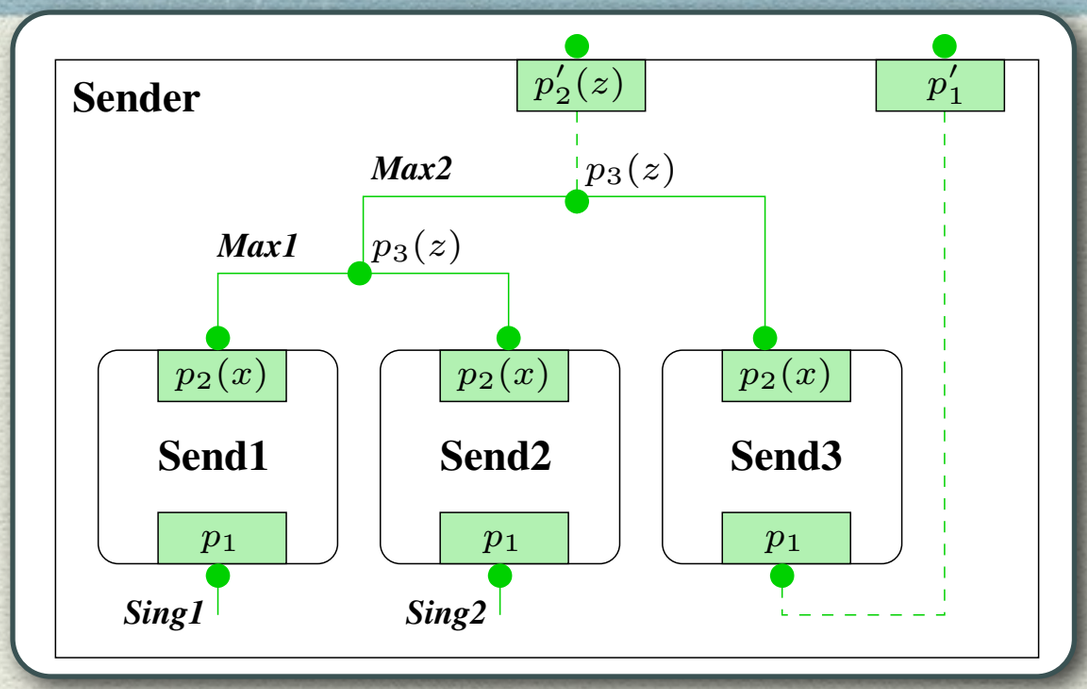
IV. Conclusion & Improvements

- ✓ Summary:
- ☑ Important properties (particularly timed) successfully verified on a real-world example
- ☑ Automatic generation of formal models out of GenoM specifications
- ☑ Manageable state spaces due to careful modeling and verification choices

IV. Conclusion & Improvements

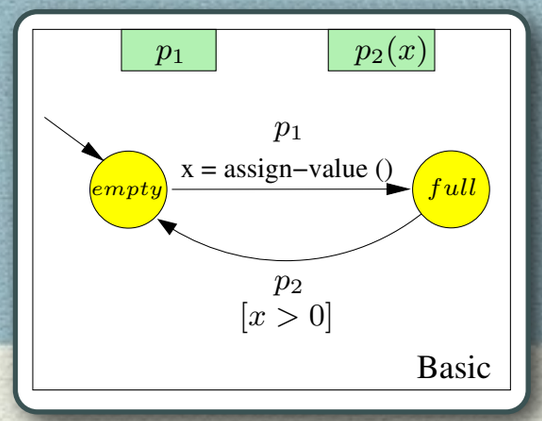
- Future work:
  - Express the properties in GenoM and translate them automatically to Fiacre
  - Automatically interpret counterexamples given by TINA to be easily understandable
  - Synthesize Fiacre models for different middleware

IV. Conclusion & Improvements



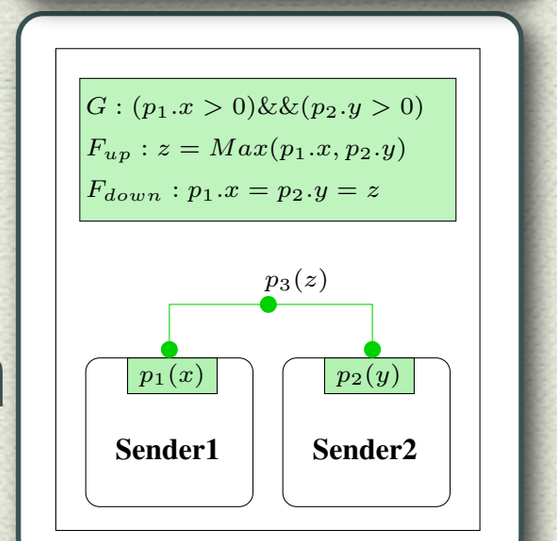

```
compound type Sender

component Basic Send1
component Basic Send2
component Basic Send3

connector Max Max1(Sender1.p2,Send2.p2)
connector Max Max2(Max1.p3,Send3.p2)
connector Singleton Sing1 (Send1.p1)
connector Singleton Sing2 (Send2.p1)

export port Intport p2' is Max2.p3
export port Intport p1' is Send3.p1

end
```

Behavior



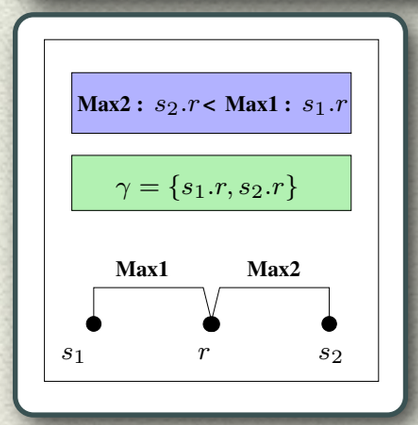

```
port type IntPort (int x)
port type ePort ()

atomic type Basic
   data int x = 0
   export port ePort p1() is p1
   export port intPort p2(x) is p2

   place empty
   place full

   initial to empty

   on p1 from empty to full
      do { x = assign-value();}
   on p2 provided [x > 0]
   from full to empty
end
```

Interaction



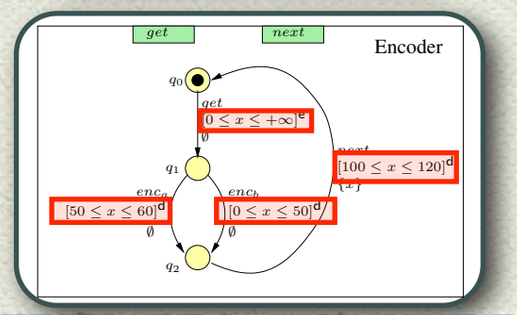

```
connector type Max (intPort p1, intPort p2)
   data int z
   define [ p1p2 ]

   on p1p2 provided (p1.x > 0) && (p2.y > 0)
      up { z = Max (p1.x , p2.y);}
      down { p1.x = p2.y = z ;}

   export port intPort p3(z)

end
```

Priority



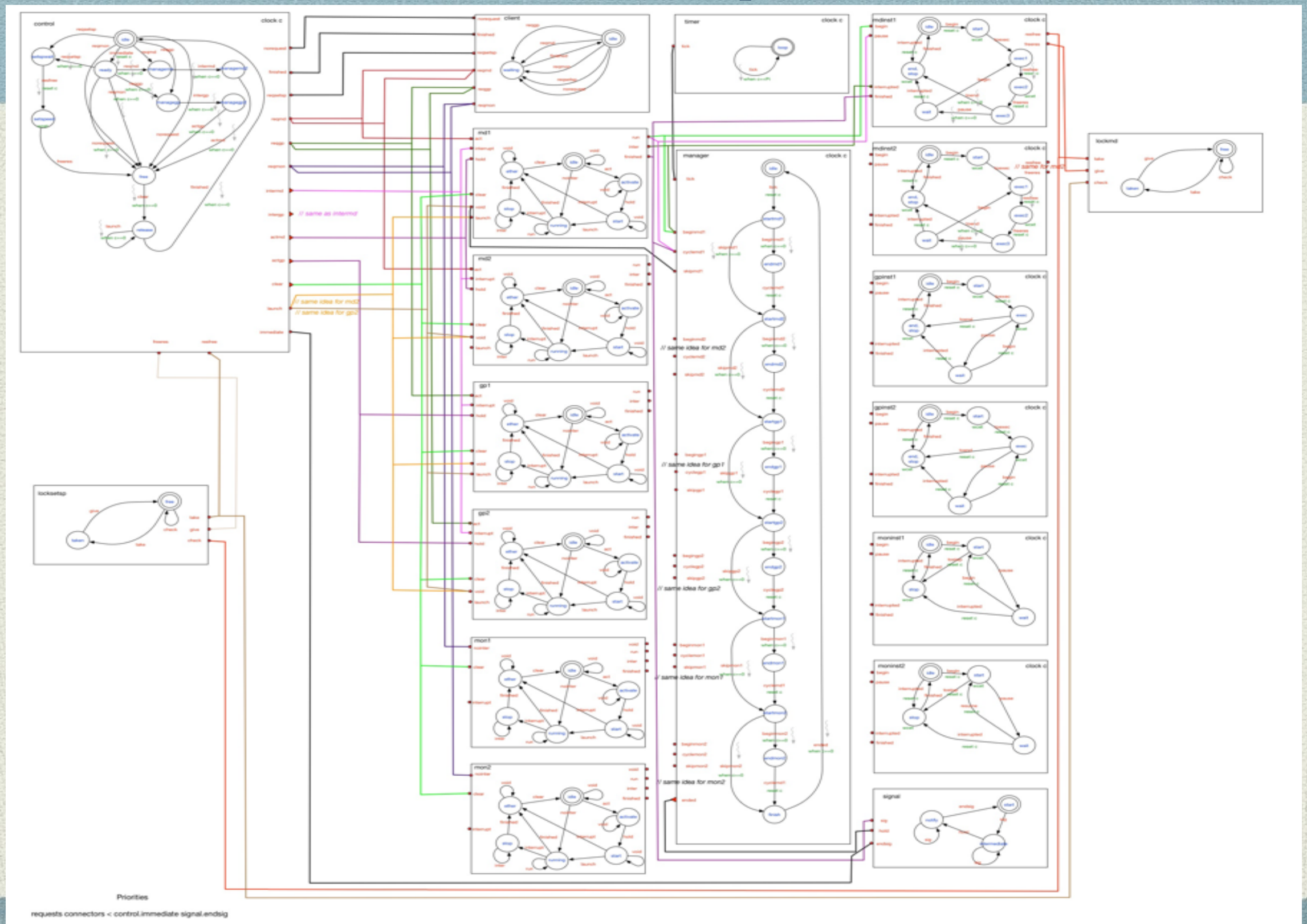

```
connector Max1 (s1, r)
connector Max2 (s2, r)

priority maximal if (s1.x > s2.x)
      Max2 < Max1
```

Time




```
atomic type Encoder
   export port intPort get
   export port intPort intPort next
   port intPort enca compute
   port intPort encb

   clock x unit millisecond

   place q0
   place q1
   place q2

   initial to q0

   on get from q0 to q1
   when x in [0,-] eager
   on enca from q1 to q2
   when x in [50,60] delayable
   on encb from q1 to q2
   when x in [0,50] delayable
   on next from q2 to q0
   when x in [100,120] delayable
   reset x
end
```

IV. Conclusion & Improvements



Special thanks to

LAAS/RIS: Félix Ingrand, Anthony Mallet
LAAS/VerTICS: Bernard Berthomieu, Silvano Dal Zilio

Part of this work is funded by the
H2020 European project CPSE Labs
under grant agreement No 644400

Thanks for your attention

–Mohammed