Validation and Verification of Time Properties of the Functional Level of Autonomous Vehicles

Félix Ingrand
(Mohammed Foughali,
Anthony Mallet)
LAAS-RIS

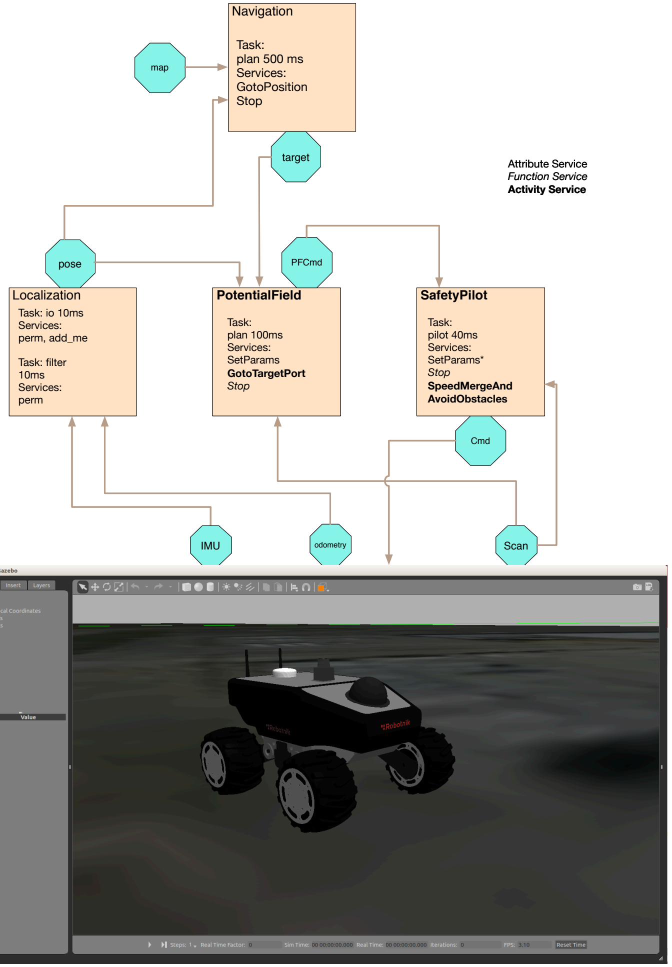
October 10, 2017

Autonomous Vehicle and Software

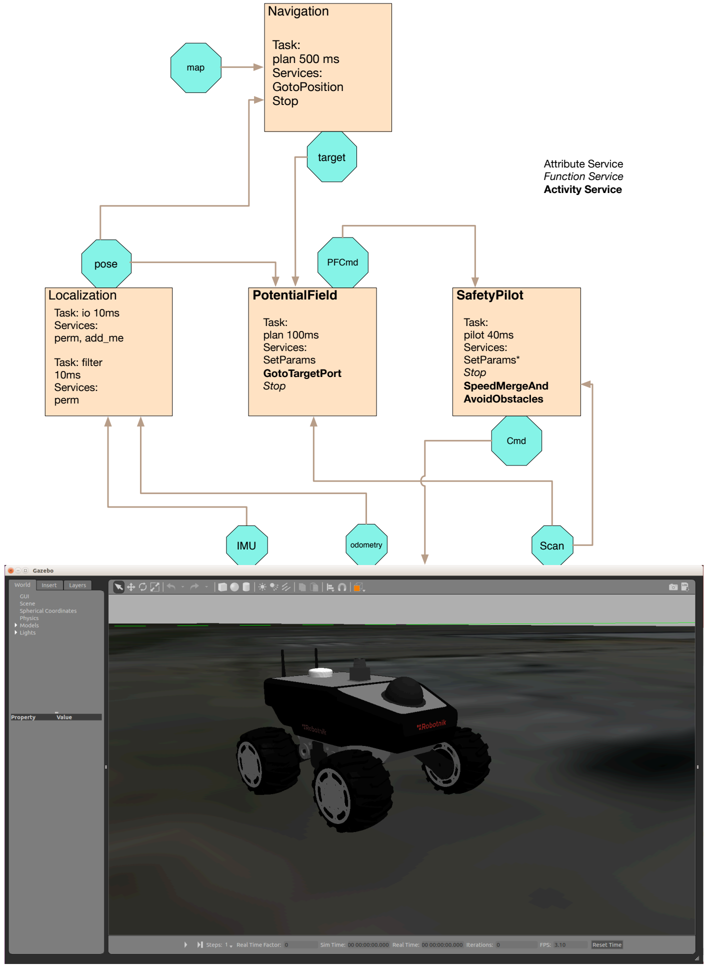
Software represents a large part of the development of Autonomous Vehicle, yet, most of it is not V&V...

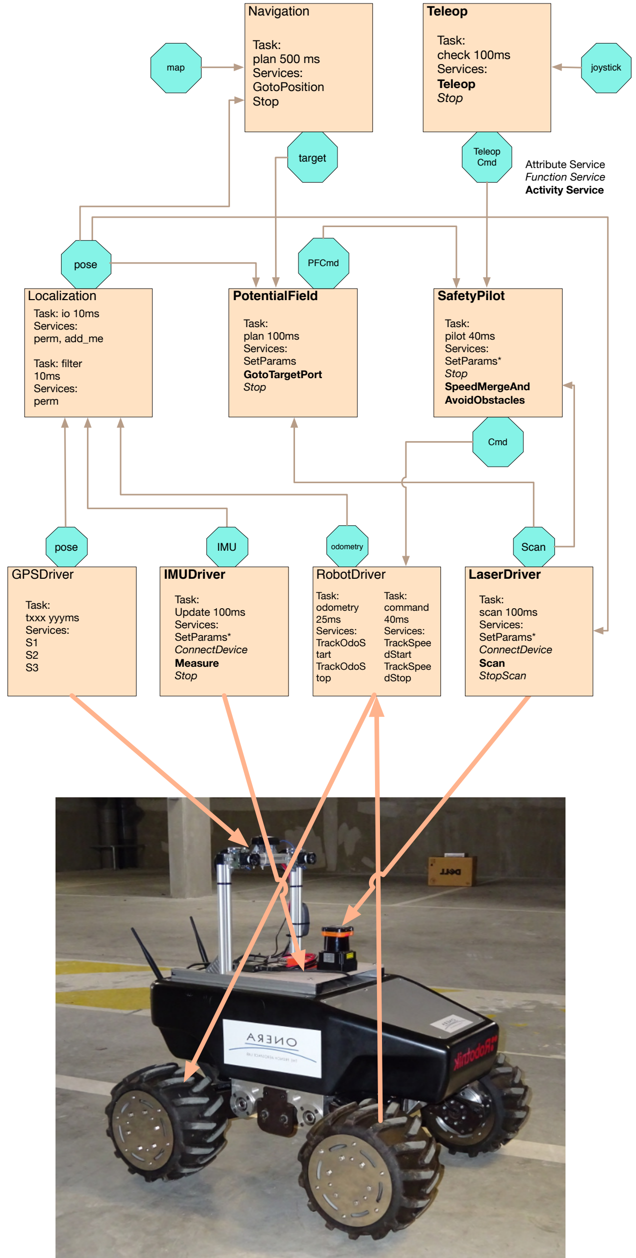
...while it is for some of these complex systems:

Robotic Software Architecture

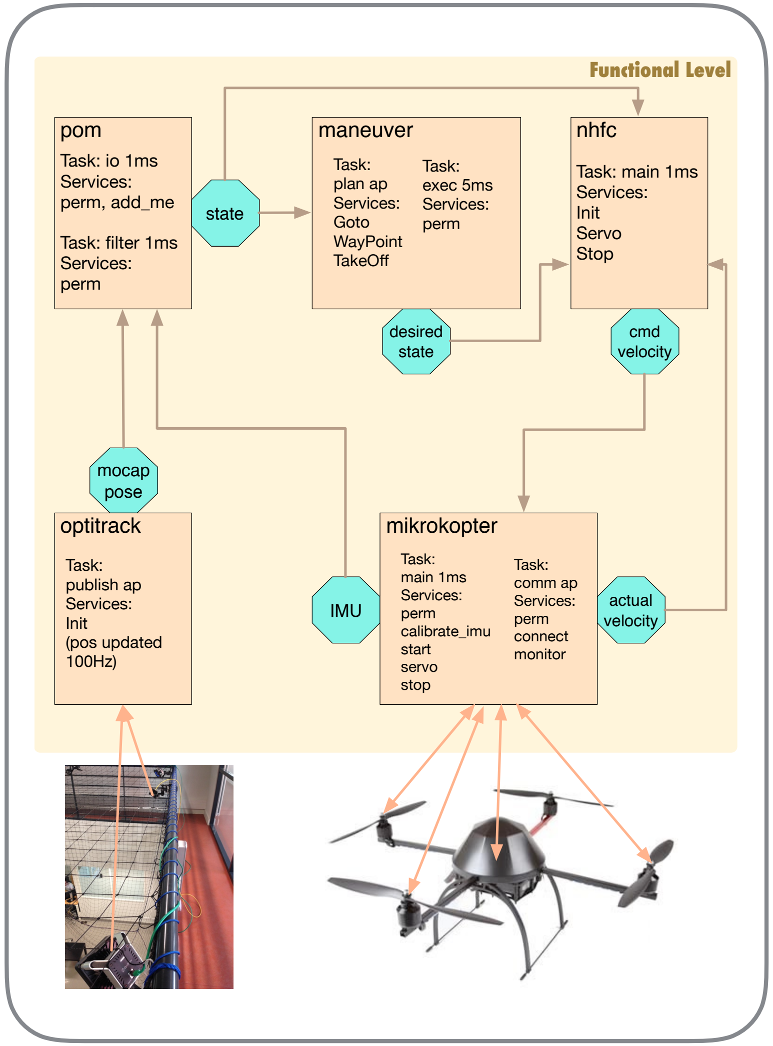


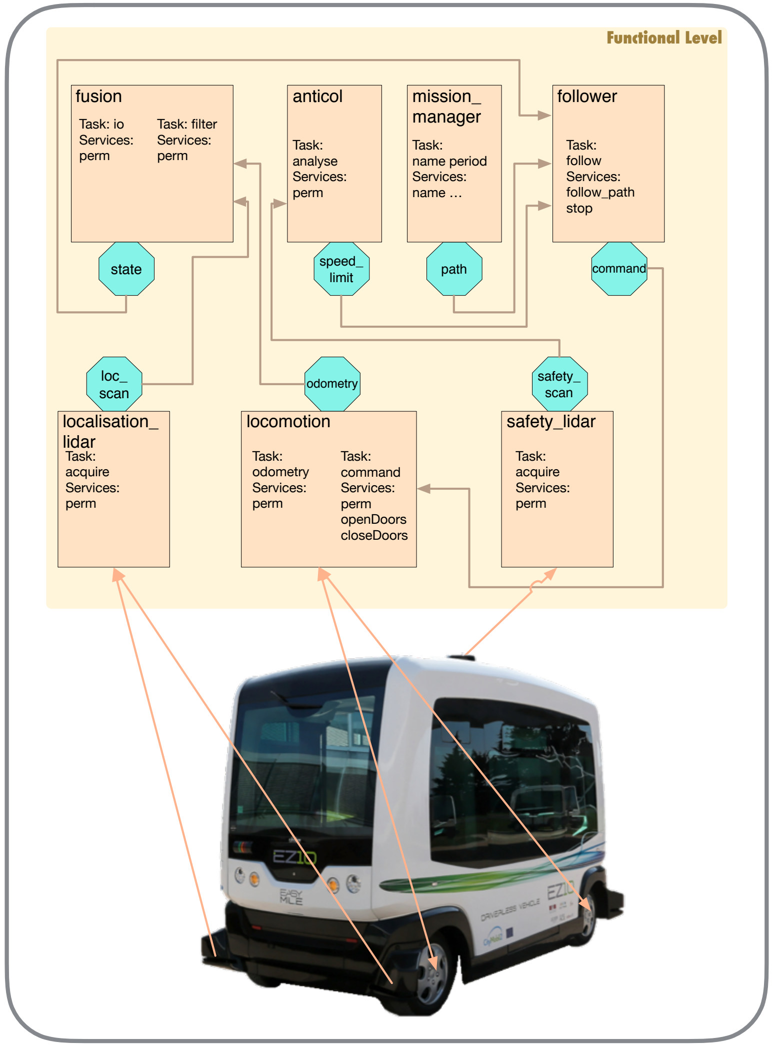
Robotic Software Architecture





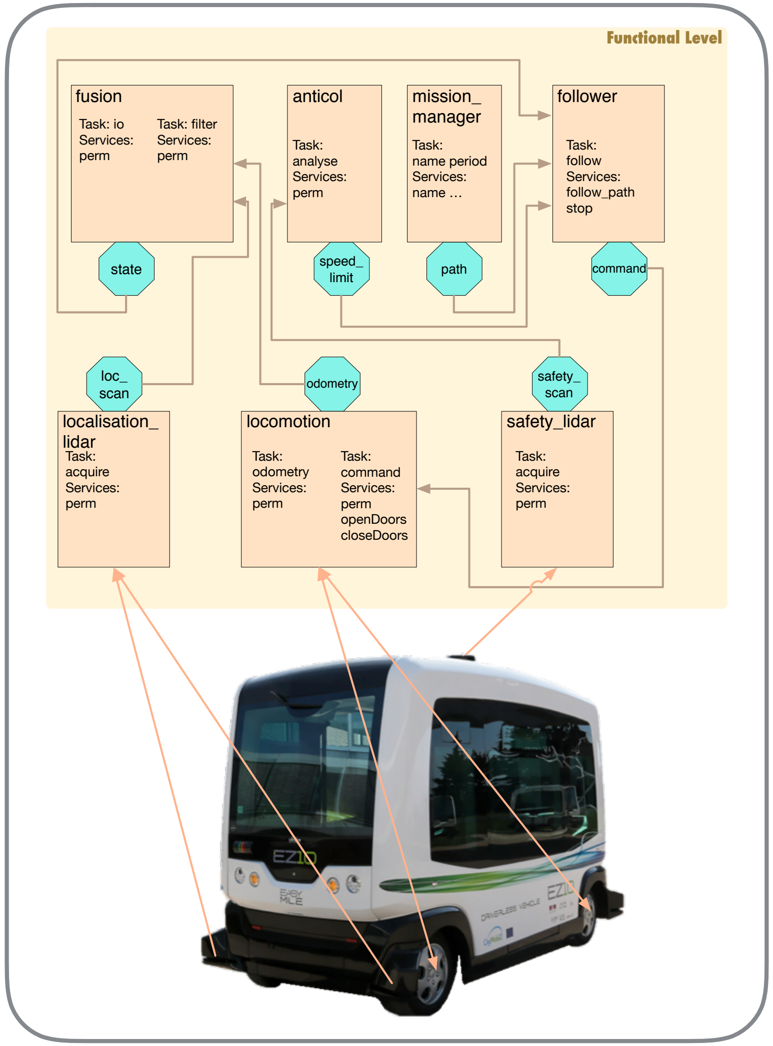
Robotic Software Architecture





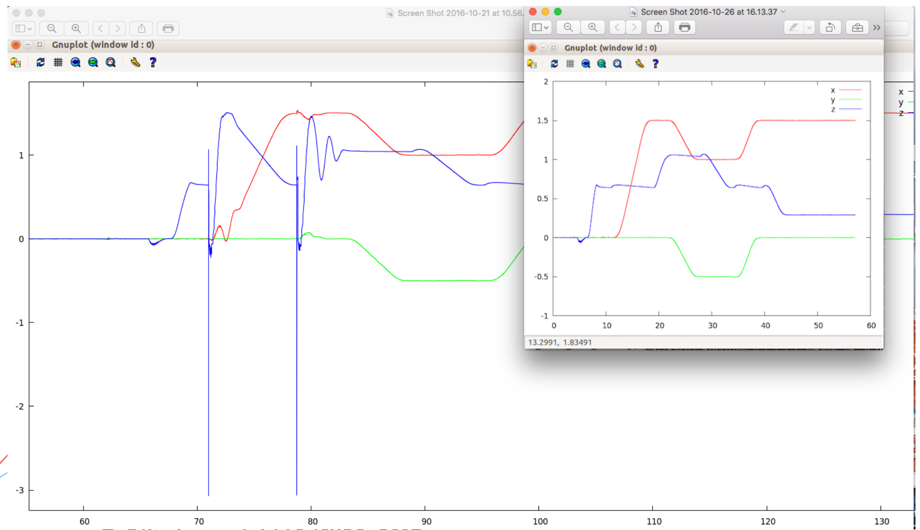
What do robotic software developers want?

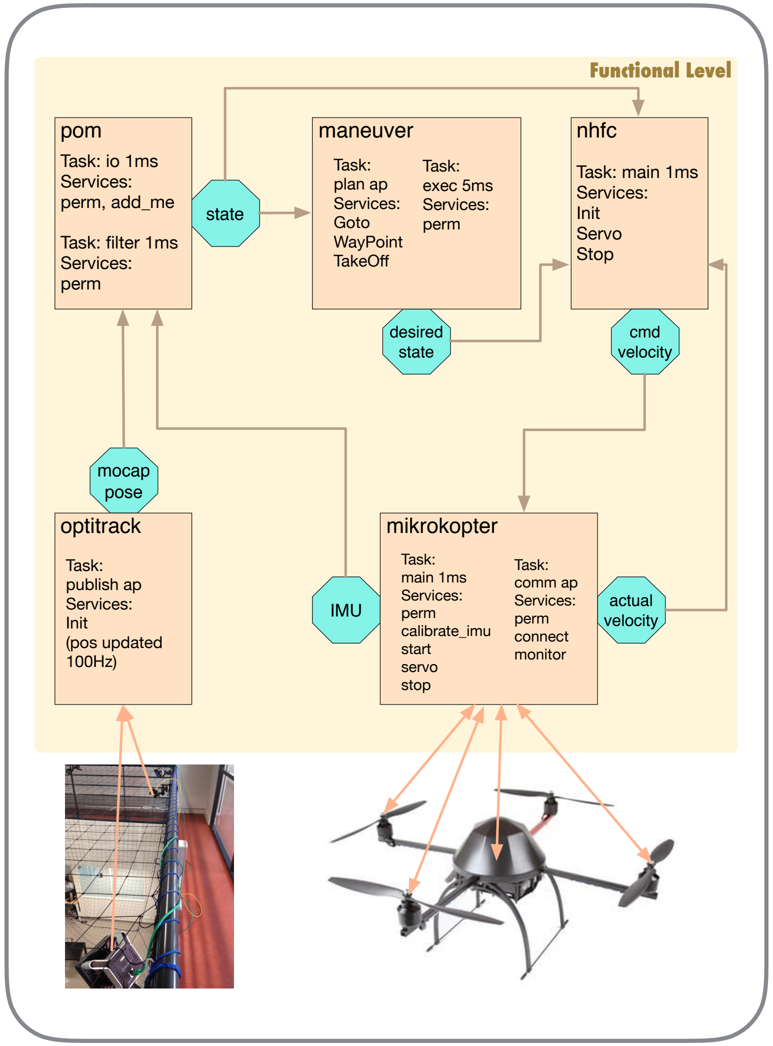
- Check that the autonomous bus drives safely
  - Stop in time when an obstacle has been detected
  - The door does no open while moving
  - Speed command is produced “timely”
  - Path following remain in bound
- Check that the robot has a consistant perception/ action loop
  - Laser scan/freq and range
  - speed control (freq and value)
  - Time for an emergency stop



What do robotic software developers want?

- Check that the drone flies safely
  - Tasks are scheduled as specified
  - Control laws are properly run
  - Propeller velocity command is produced "timely"
  - Consistant perception/action loop
  - Localisation properly "merged"
  - Time taken for an emergency stop (hover in the current place)
  - No deadlock upon start



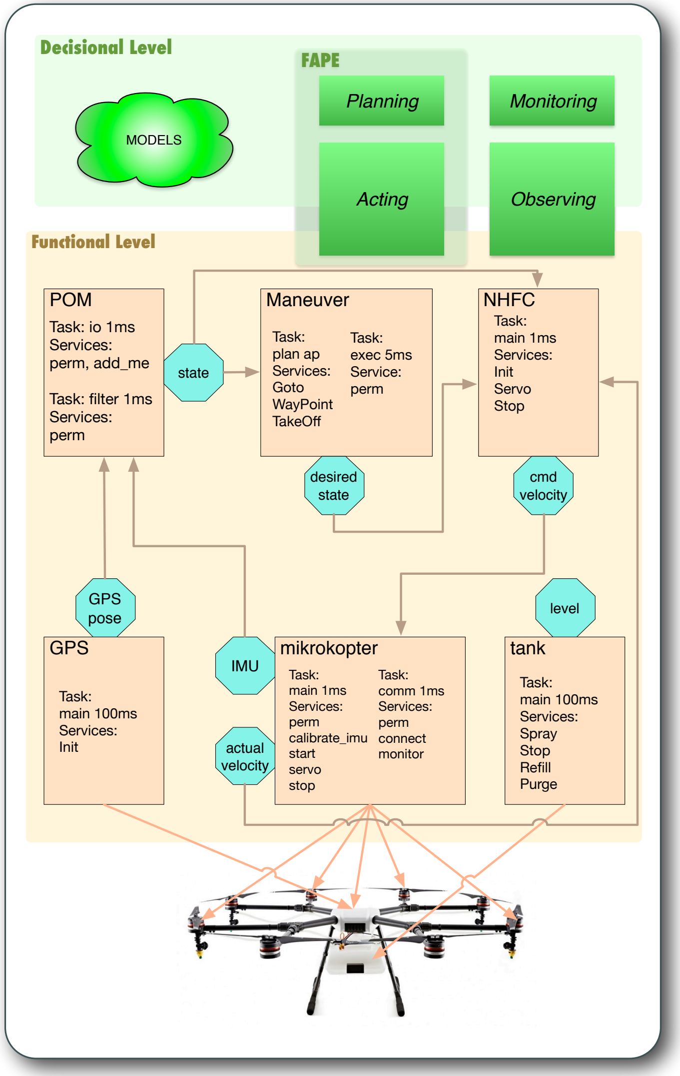


Software Validation and Verification

- Require formal models and “checking” techniques
  - either with models directly “verifiable” (e.g., Petri nets, timed automata, etc)
  - or with models which can be obtained or “translated” from specifications

Different Situations w.r.t. Using Formal Models on Autonomous Robots

- Already model based (decisional : planning, monitoring, FDIR, observing)
- Directly using formal framework (e.g. synchronous approaches: Esterel/Lustre/Signal) (Mihaela's talk)
- No model at all (checking the code after "formalizing" what it does or the properties it should satisfy)
- Partial models (software engineering models: e.g. GenoM; specification models: UML, RobotML, etc; components based: BCM)



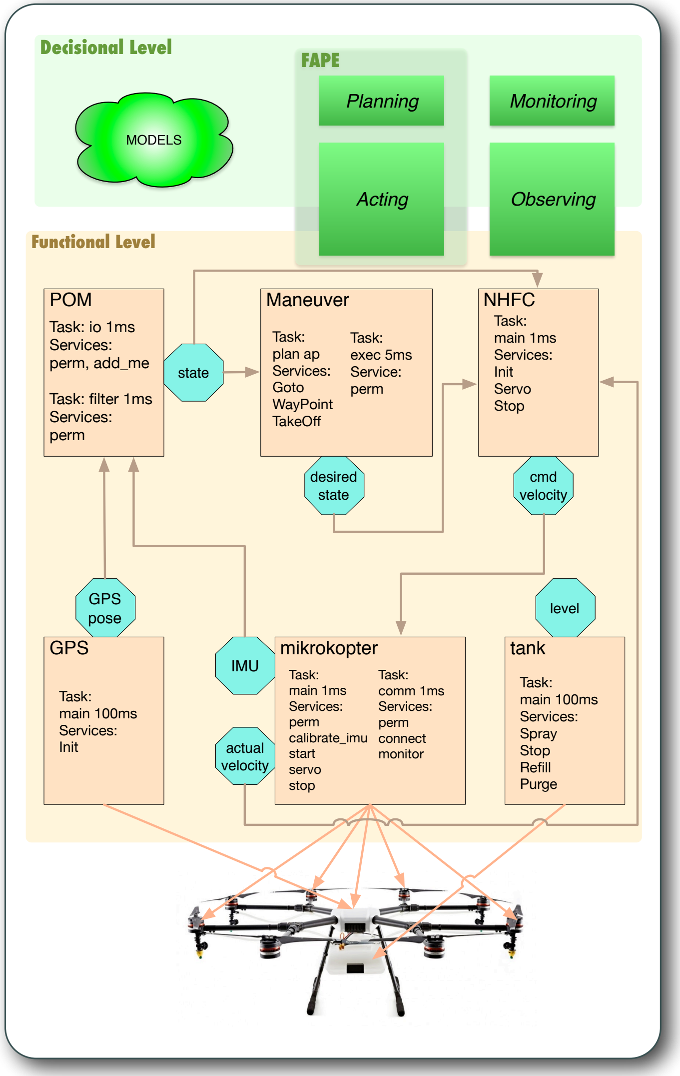
Different Situations w.r.t. Using Formal Models on Autonomous Robots

Already model based (decisional : planning, monitoring, FDIR, observing)

Directly using formal framework (e.g. synchronous approaches: Esterel/Lustre/Signal) (Mihaela's talk)

No model at all (checking the code after "formalizing" what it does or the properties it should satisfy)

Partial models (software engineering models: e.g. GenoM; specification models: UML, RobotML, etc; components based: BCM)



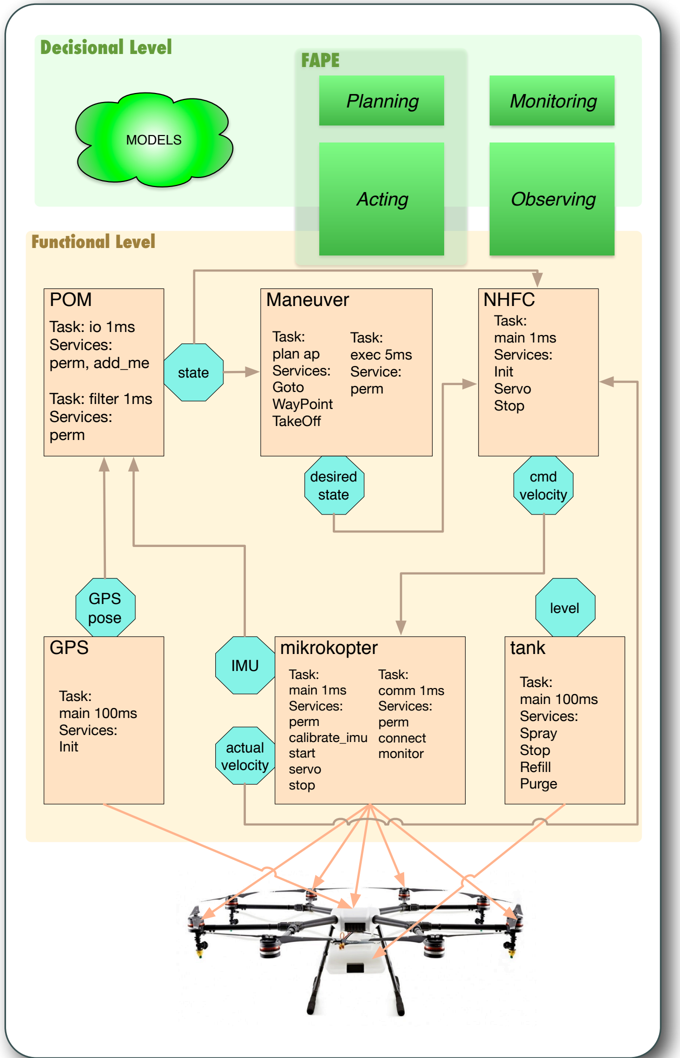
Our Approach

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

Model-Driven Software Engineering

- BIP (Verimag)
- Fiacre/TINA (LAAS/VerTICS)
- UPPAAL (Uppsala & Aalborg University)

Formal Methods/ Frameworks



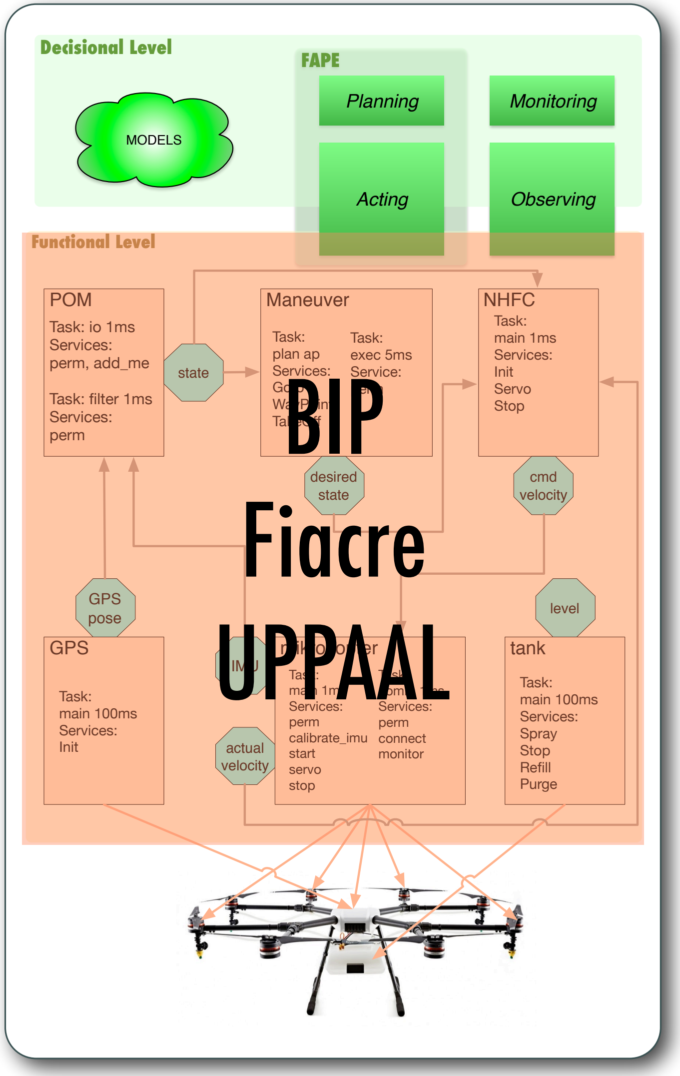
Our Approach

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

Model-Driven Software Engineering

- BIP (Verimag)
- Fiacre/TINA (LAAS/VerTICS)
- UPPAAL (Uppsala & Aalborg University)

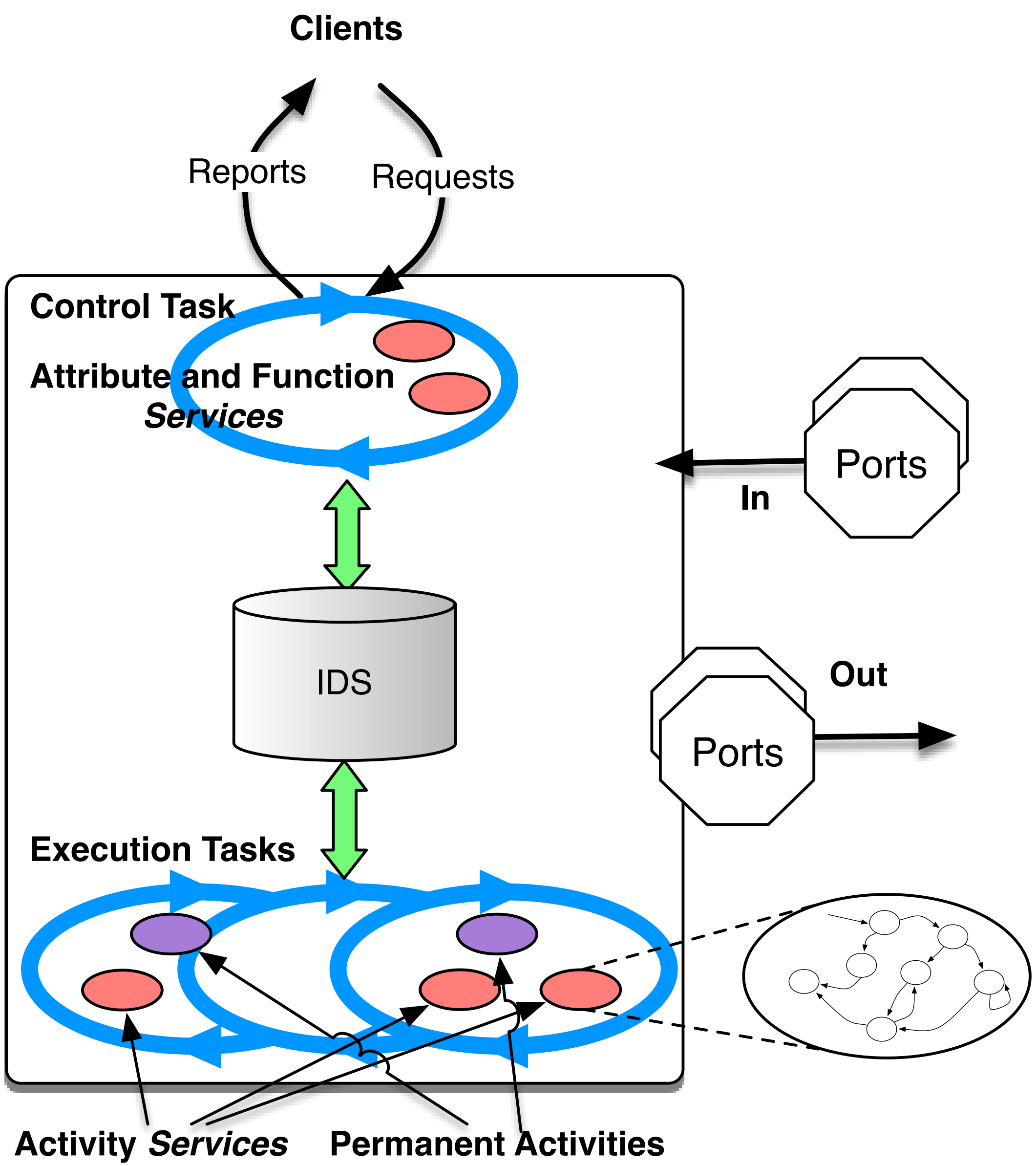
Formal Methods/ Frameworks



LAAS CNRS

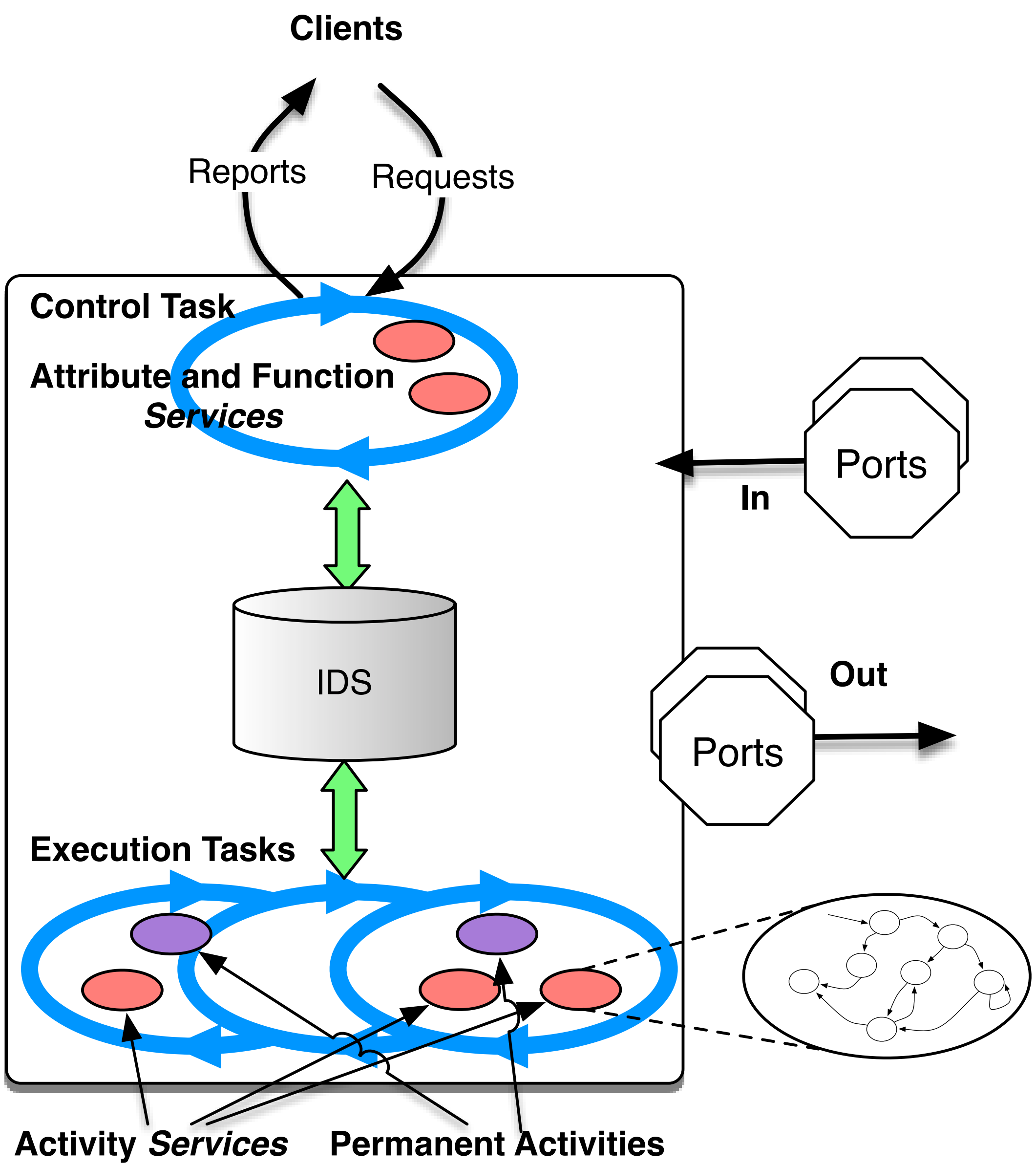
GenoM

- To design a typical generic module which will be instantiated according to each specific module
- a module is a program
- a module has I/O
    - control: requests to start services/reports their results
    - data: ports in(to read external data) and out (to write external data)
- it supports a cyclic control task (aperiodic)
- and one or more cyclic tasks, periodic or aperiodic
- it provides services (fast and slow) to which we will associate C/C++ code
    - in the control task
    - and the executions task(s)
- the slow one can have different steps (automata) to perform their processing
- services share a common data structure for the need of their computation (parameters, computed values, internal state variables, etc)
- execution tasks may have a permanent activity
- there may be exceptions, and incompatible services (they cannot run at the same time)



GenoM

- Specify components
  - IDS
  - Ports
  - Tasks
  - Services
    - Attribute, function and activity (automata)
    - and attached codels…



GenoM .gen & codels (example)

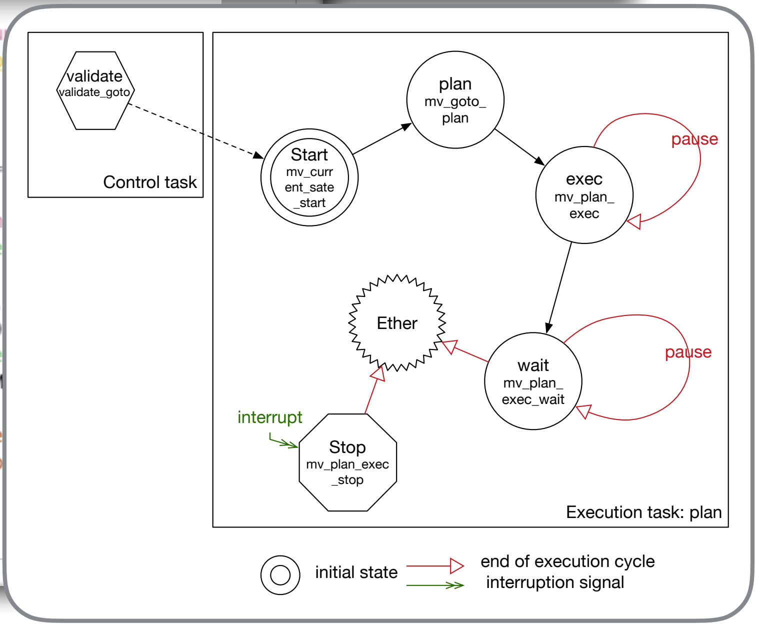
GenoM models: .idl & .gen

- IDS
- Services (automata)
- Ports
- Tasks
- Exceptions
- etc

```
/* ---- Data structures and IDS ---- */

ids {
  demo::state    state;        /* Current state */
  demo::speed    speedRef;     /* Speed reference */
  double         posRef;
};
```

```
/* ---- Posters declarations ---- */

port out demo::state Mobile;
```

```
/* ---- Execution task declaration ---- */

  priority      100;
  stack         4000;
  codel <start> InitDemoSDI(out ::ids, port out Mobile) yield ether;
};
```



GenoM .gen & codels (example)

GenoM models: .idl & .gen

- IDS
- Services (automata)
- Ports
- Tasks
- Exceptions
- etc

```
module demo {

  const unsigned long task_period = 400;
  const double millisecond = 0.001;
```

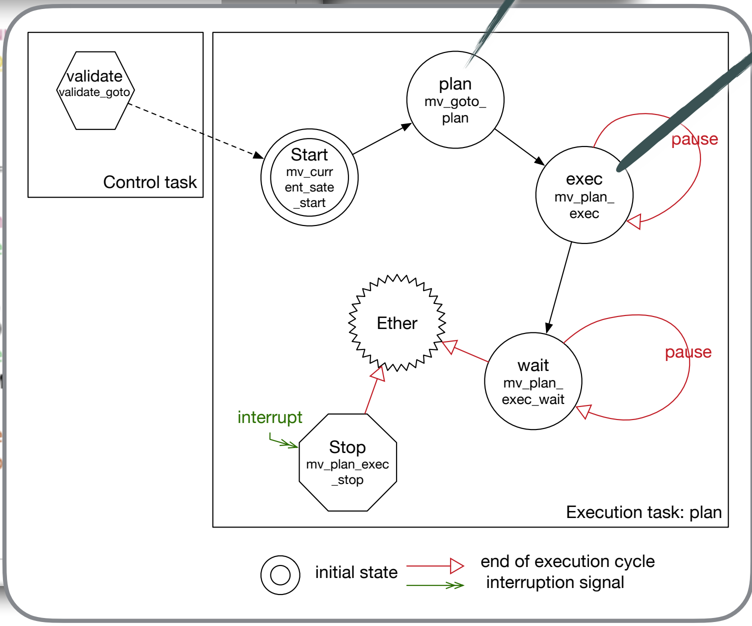
```
/* ---- Data structures and IDS ---- */

ids {
  demo::state    state;        /* Current state */
  demo::speed    speedRef;     /* Speed reference */
  double         posRef;
};
```

```
/* ---- Posters declarations ---- */

port out demo::state Mobile;
```

```
attribute SetSpeed(in speedRef = demo::SLOW
{
  doc          "To change speed";
  validate     controlSpeed (local in speedRef);
  throw        INVALID_SPEED;
};
```

```
/* ---- Execution task declaration ---- */

  priority      100;
  stack                4000;
  codel <start> InitDemoSDI(out ::ids, port out Mobile) yield ether;
};
```

```
  doc          "Stops motion and interru
  interrupts   MoveDistance, GotoPositio
};

  doc          "Move of the given distan
  validate     controlDistance(in distRe

  codel <start> mdStartEngine(in distRef,
                              out posRef)
  codel <exec>  mdGotoPosition(in speedRe
                              port out M
  codel <end, stop>
                mdStopEngine() yield ethe
  interrupts    MoveDistance, GotoPositio
  task          motion;
  throw TOO_FAR_AWAY;
};
```




```
/* --- Activity GotoPosition  and Monitor ------------------------- */

/** Validation codel controlPosition of activity GotoPosition
 *   and Monitor.
 *
 * Returns ok.
 * Throws TOO_FAR_AWAY.
 */
demo_event
controlPosition(const double *posRef)
{
  if (*posRef > DEMO_MACHINE_LENGTH/2 ||
      *posRef < -DEMO_MACHINE_LENGTH/2) {
    return demo_TOO_FAR_AWAY;
  }
  return demo_ok;
}
```

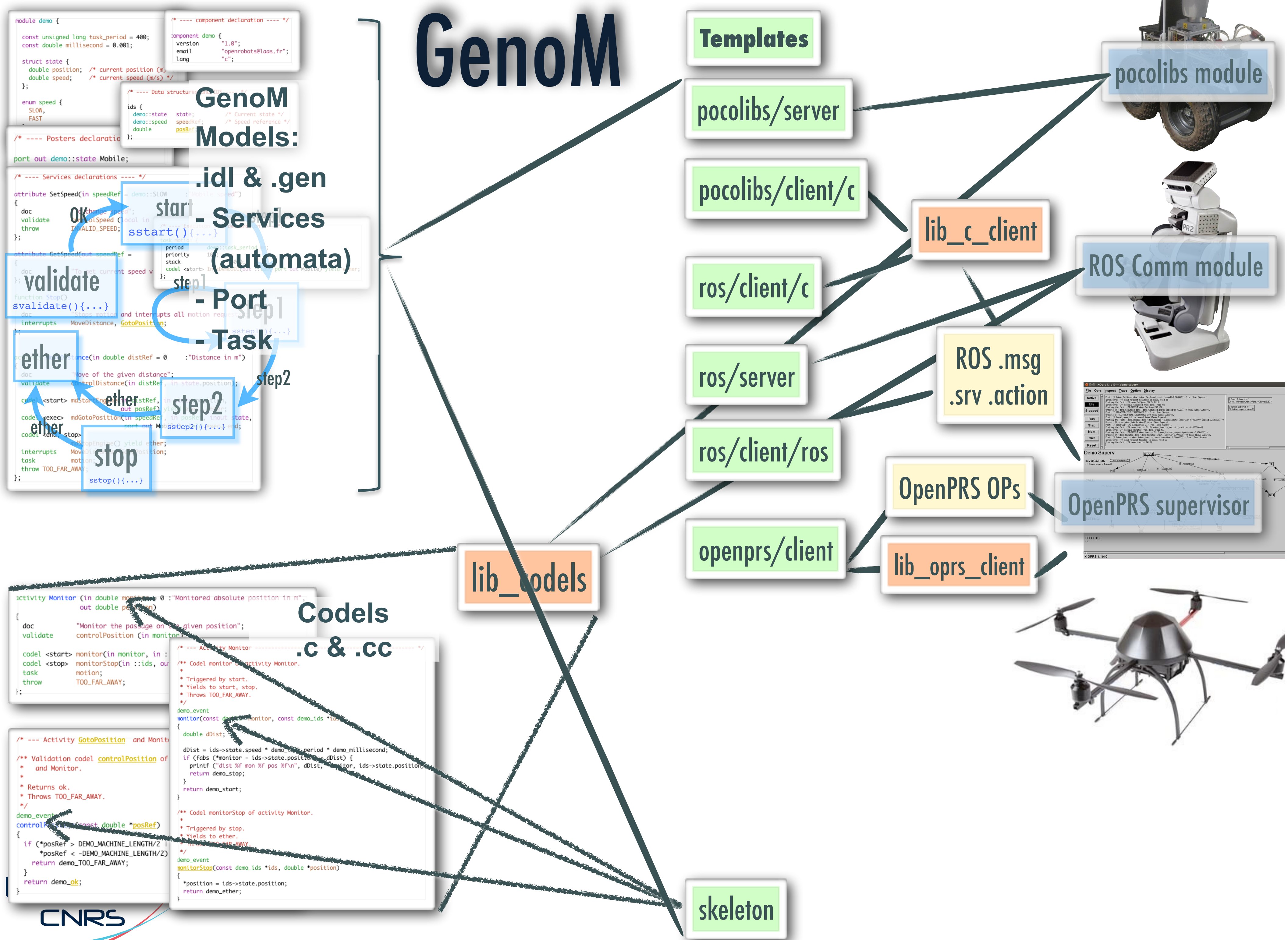
Codels .c & .cc

WCET

```
/* --- Activity Monitor ---------------------------------------------- */

/** Codel monitor of activity Monitor.
 *
 * Triggered by start.
 * Yields to start, stop.
 * Throws TOO_FAR_AWAY.
 */
demo_event
monitor(const double *monitor, const demo_ids *ids)
{
  double dDist;

  dDist = ids->state.speed * demo_task_period * demo_millisecond;
  if (fabs (*monitor - ids->state.position) < dDist) {
    printf ("dist %f mon %f pos %f\n", dDist, *monitor, ids->state.position);
    return demo_stop;
  }
  return demo_start;
}

/** Codel monitorStop of activity Monitor.
 *
 * Triggered by stop.
 * Yields to ether.
 * Throws TOO_FAR_AWAY.
 */
demo_event
monitorStop(const demo_ids *ids, double *position)
{
  *position = ids->state.position;
  return demo_ether;
}
```

Why are GenoM specifications good for V&V?

- Codel granularity (better parallelism specification)
- Internal and external shared data access is fully specified (ports, IDS, and nothing else)
- Automata specification provides execution sequence and time/period management
- Task are clearly specified (how many, periodic, sporadic)
- Template mechanism...

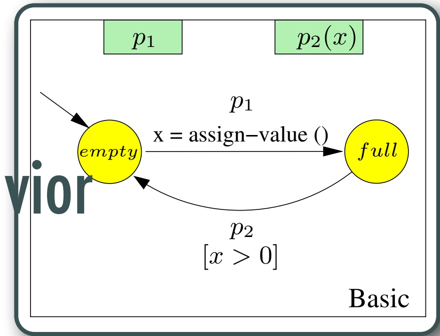

GenoM
GenoM Models:
.idl & .gen
- Services (automata)
- Port
- Task
start
validate
step1
step2
stop
ether
OK
Codels .c & .cc
Templates
pocolibs/server
pocolibs/client/c
ros/client/c
ros/server
ros/client/ros
openprs/client
skeleton
lib_c_client
lib_codels
ROS .msg .srv .action
OpenPRS OPs
lib_oprs_client
pocolibs module
ROS Comm module
OpenPRS supervisor

Three V&V frameworks

- BIP (Verimag)
- TINA/Fiacre (LAAS/Vertics)
- UPPAAL (Uppsala & Aalborg University)

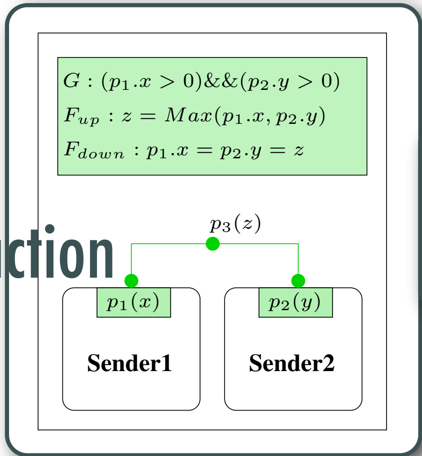
BIP Model example

Behavior



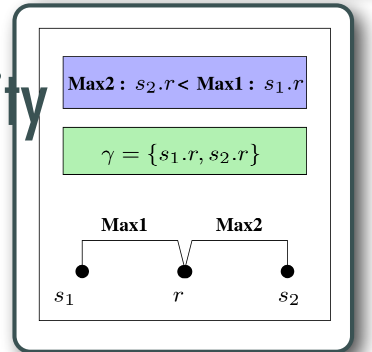

```
port type IntPort (int x)
port type ePort ()

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



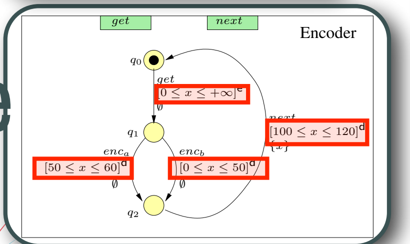

```
connector type Max (intPort p1, intPort p2)
   data int z
   define [ p1p2 ]

   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

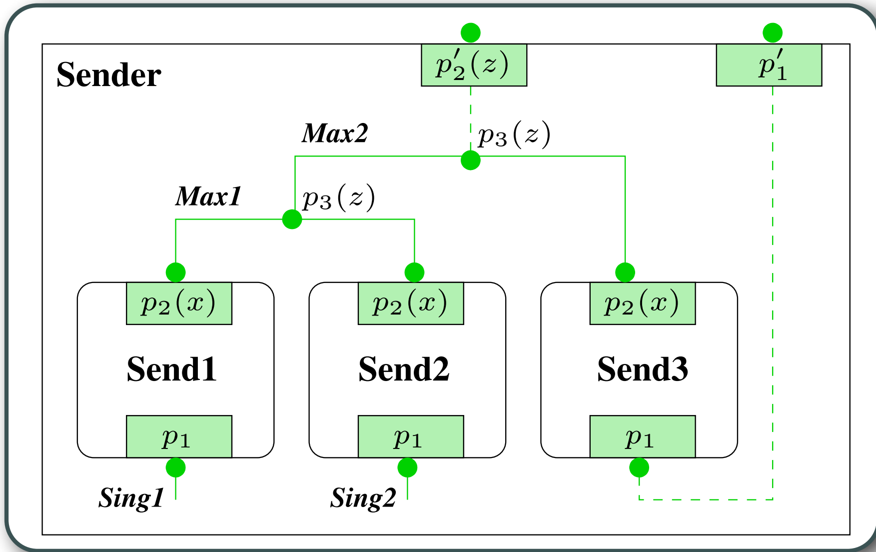



```
connector Max1 (s1, r)
connector Max2 (s2, r)

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

Time




```
atomic type Encoder
   export port intPort get
   export port intPort intPort next
   port intPort enc_a compute
   port intPort enc_b

   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 enc_a from q1 to q2
   when x in [50,60] delayable
   on enc_b 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
```



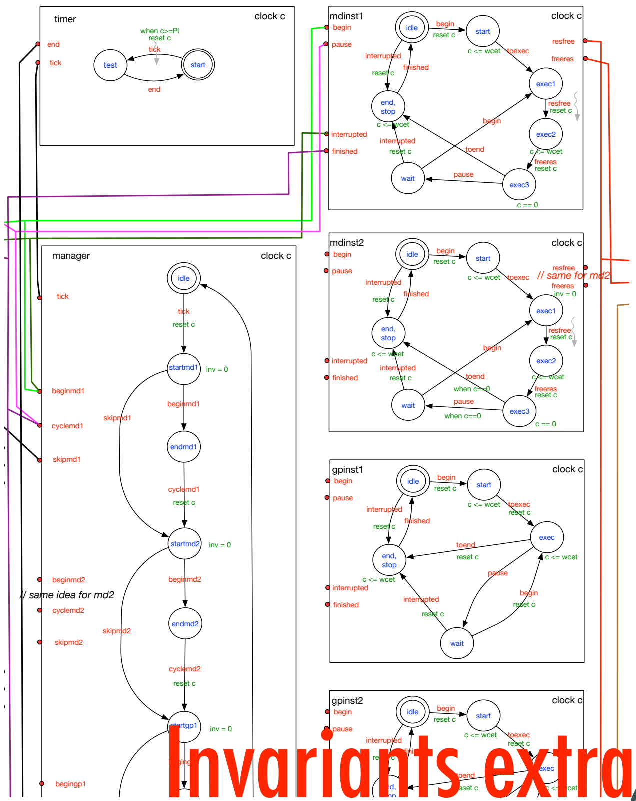

```
compound type Sender

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 p'2 is Max2.p3
export port Intport p'1 is Send3.p1

end
```

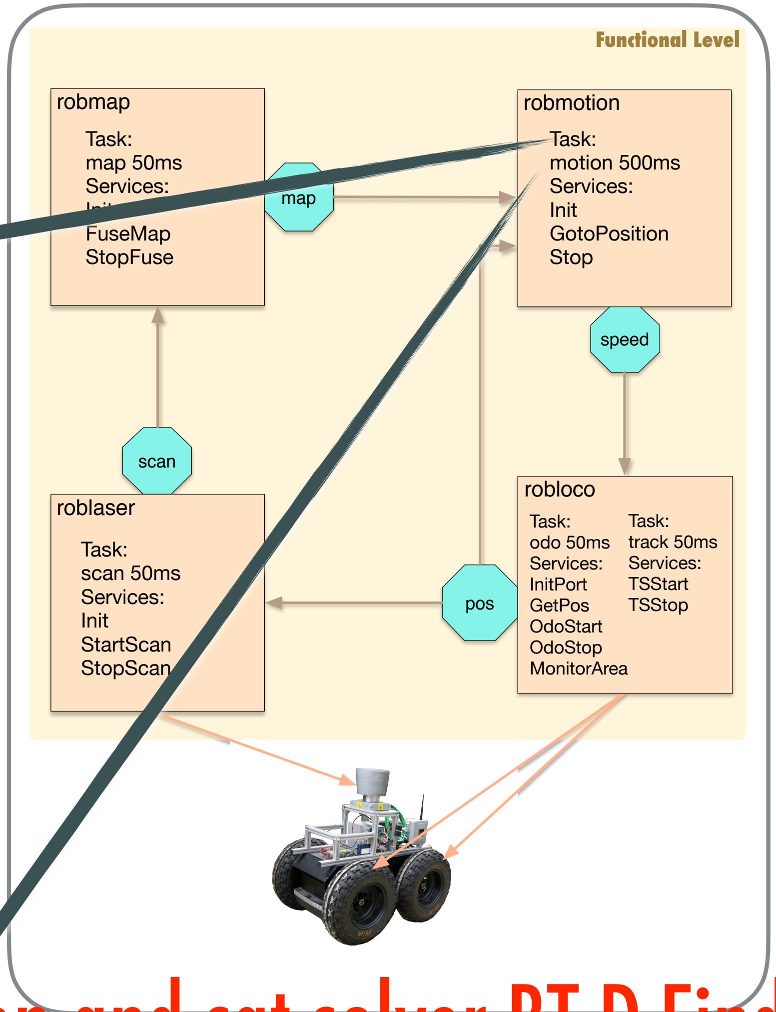
GenoM to BIP

A template that produces the BIP model of any GenoM specification for the PocoLibs implementation

example:

```
/* plan timer */

atom type TIMER_plan_robmotion()
        clock c unit millisecond
        export port Port tick()
        place loop
        initial to loop

        on tick
        from loop to loop
        provided (c>=500.0)
        do { c = 0; }
end
```



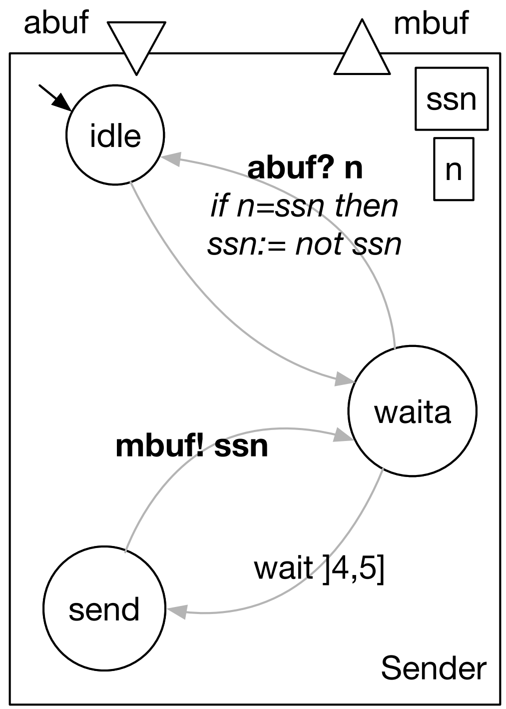


Invariants extraction and sat solver RT D-Finder

Runtime Checking with the RT BIP Engine

Fiacre example

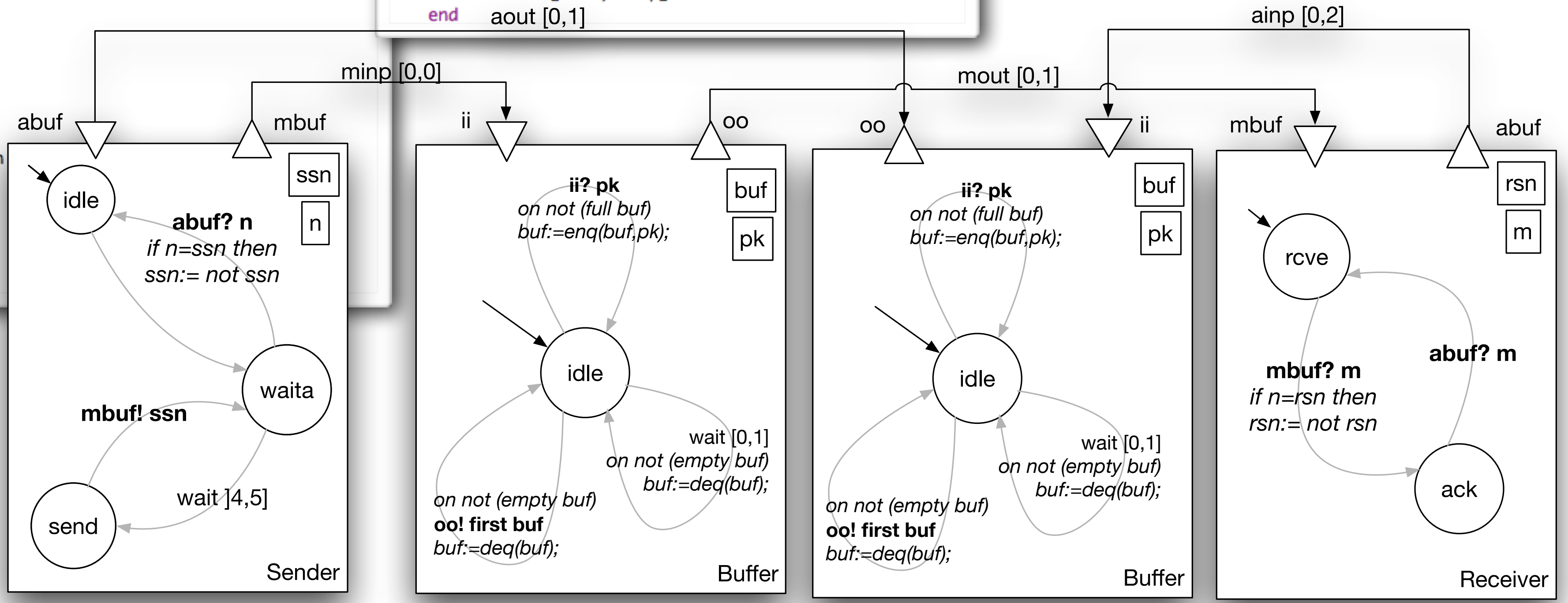
```
process sender [mbuff: out packet, abuff: in packet] is
    states idle, send, waita
    var ssn, n: seqno := false  // ssn is current sequence number
    from idle
        /* should also retrieve data from user */
        to waita
    from send
        mbuff! ssn;
        to waita
    from waita
      select
         abuff? n;
         if n = ssn
         then ssn := not ssn
         end;
         to idle
        [] wait ]4,5];
         /* resend */
         to send
      end
```



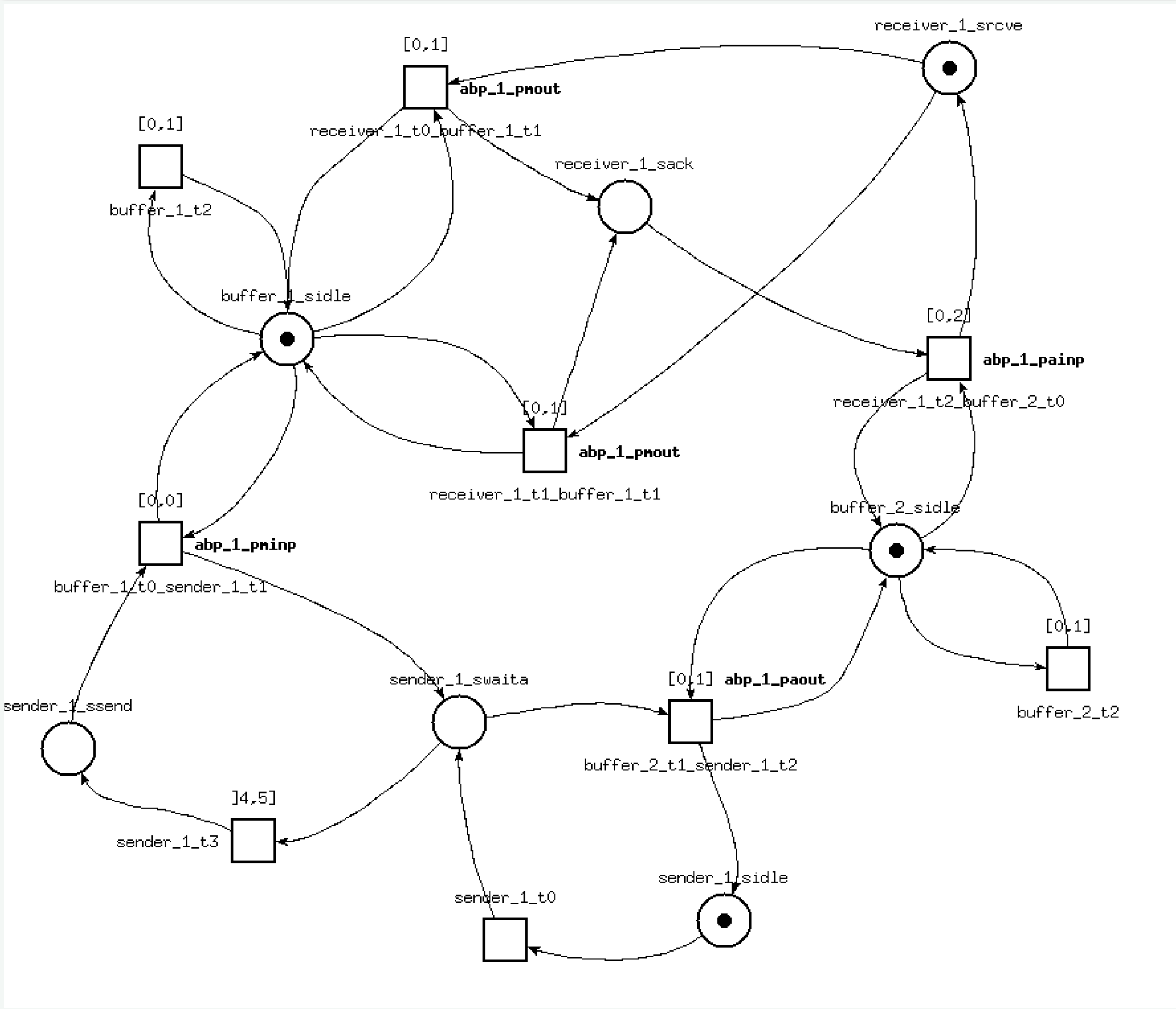
Fiacre Model example: Alternate Bit Protocol

```
/* Processes */

process buffer [ii: in packet, oo: out packet] is
    states idle
    var buff : queue 1 of packet := {||},
        pkt: packet
    from idle
    select
       /* getting new packet */
       ii?pkt;
       on not (full buff);  // should be redundant but prevents
                            // queue exception if time-out too small
       buff := enqueue (buff,pkt);
       to idle
    [] /* putting first packet */
       on not (empty buff);
       oo!first buff;
       buff := dequeue buff;
       to idle
    [] /* losing a packet */
       wait [0,1];
       on not (empty buff);
       buff := dequeue buff;
       #lost;
       to idle
    end

process sender [mbuff: out packet, abuff: in packet] is
    states idle, send, waita
    var ssn, n: seqno := false  // ssn is current sequence number
    from idle
       /* should also retrieve data from user */
       to waita
    from send
       mbuff! ssn;
       to waita
    from waita
      select
        abuff? n;
        if n = ssn
        then ssn := not ssn
        end;
        to idle
      [] wait ]4,5];
        /* resend */
        to send
      end
```

```
process receiver [mbuff: in packet, abuff: out packet] is
    states rcve, ack
    var rsn: seqno := false, m: packet := true
      // rsn is expected sequence number
    from rcve
        mbuff? m;
        if m = rsn then
           /* also should deliver data to user */
           rsn := not rsn;
           to ack
        else
           // reject duplicate
           to ack
        end
    from ack
        abuff! m;
        to rcve


/* Main component */

component abp is
    port minp : packet in [0,0],
         mout : packet in [0,1],
         ainp : packet in [0,2],
         aout : packet in [0,1]
    par * in
       sender [minp, aout]
    || buffer [minp, mout]
    || buffer [ainp, aout]
    || receiver [mout, ainp]
    end
```



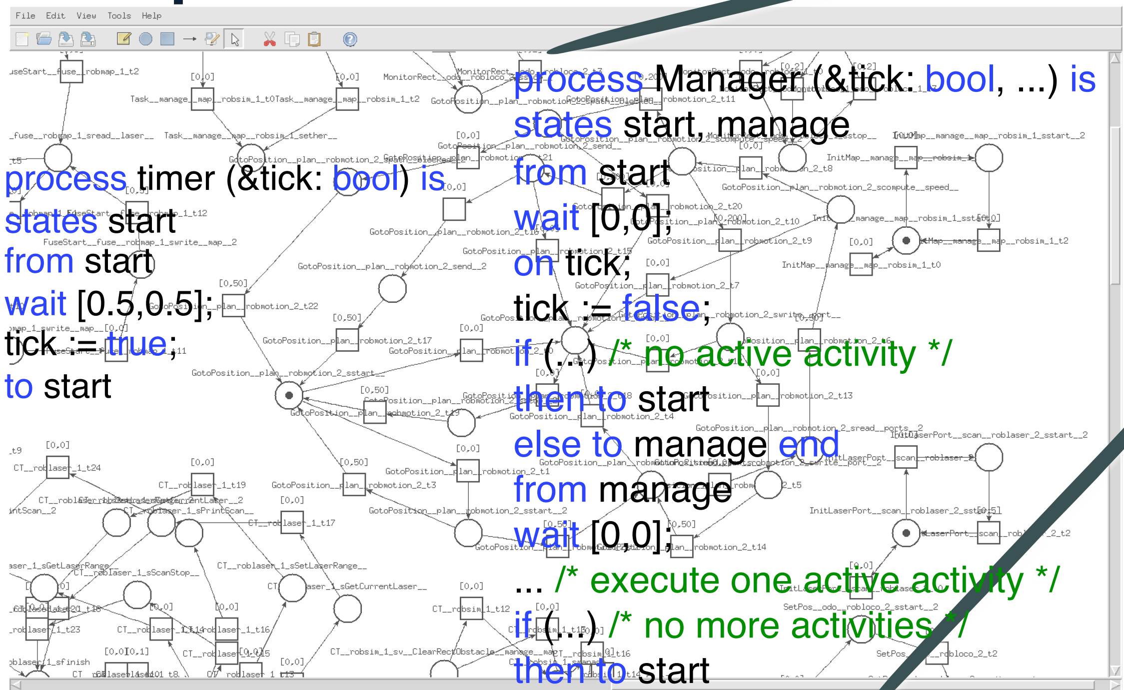
ABP FIACRE example automatically translated to Time Petri Net (TINA)



GenoM to Fiacre

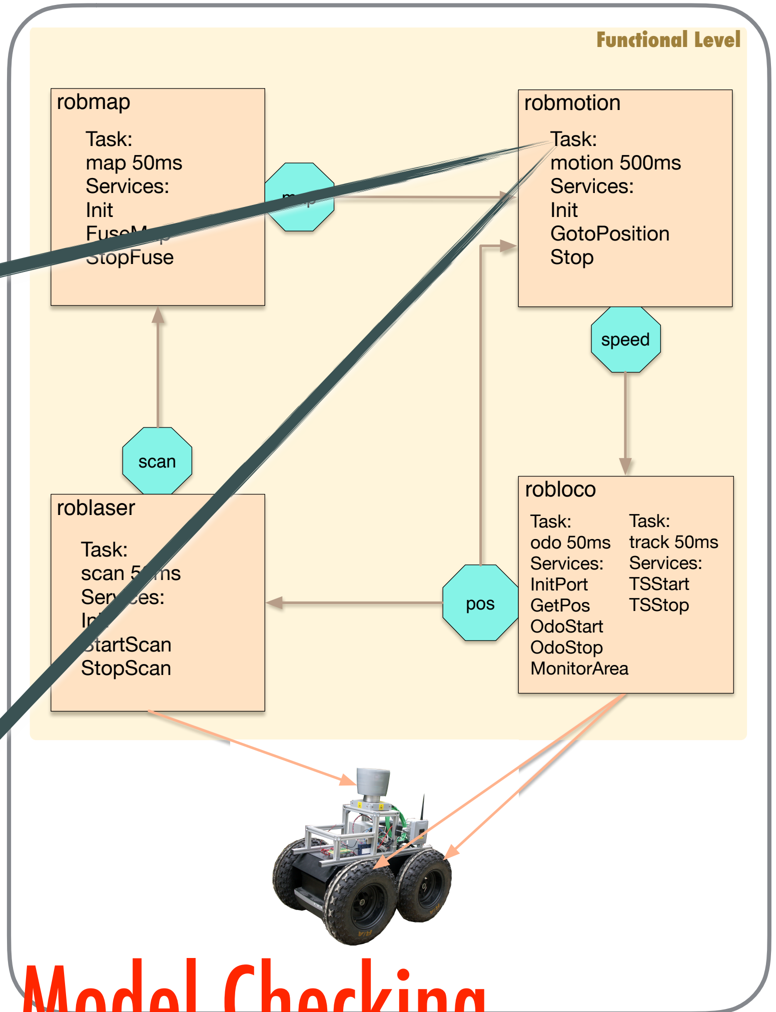
A template that produces the Fiacre model of any GenoM specification for the PocoLibs implementation

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
```



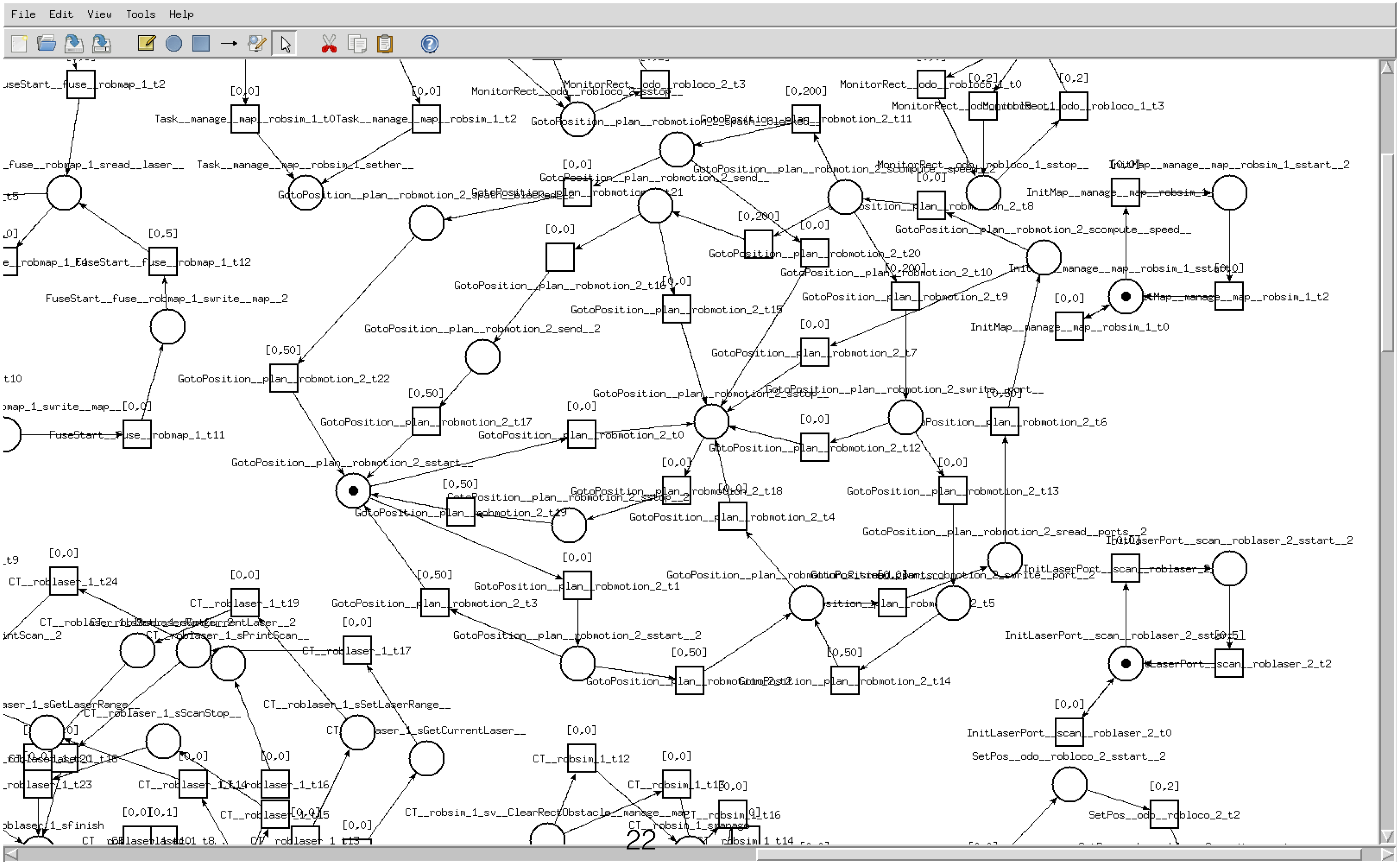
Model Checking
(on the TPN equivalent model)

LAAS CNRS

TINA
(GotoPosition Service Automata)

The complete model for RobMotion

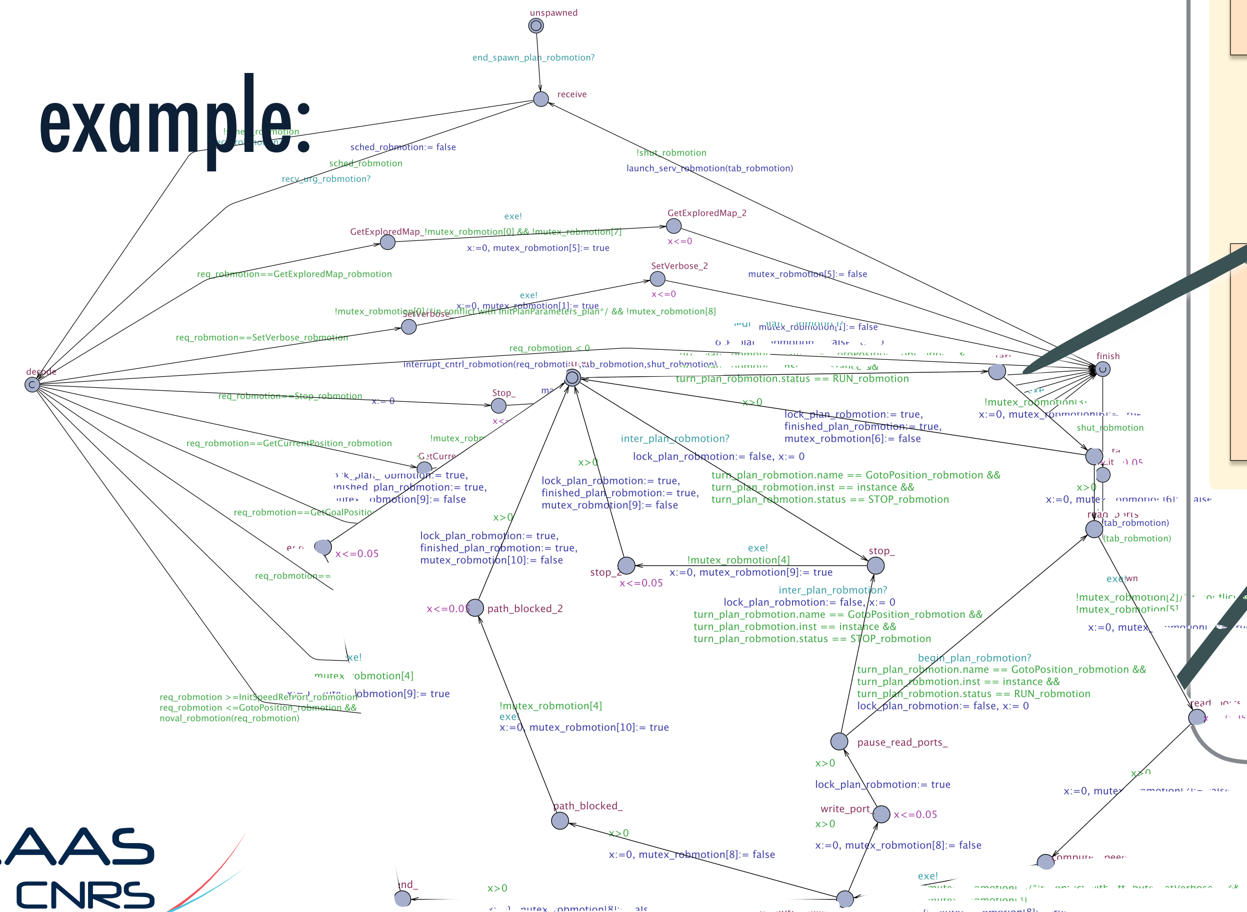
- 250 places
- 506 transitions

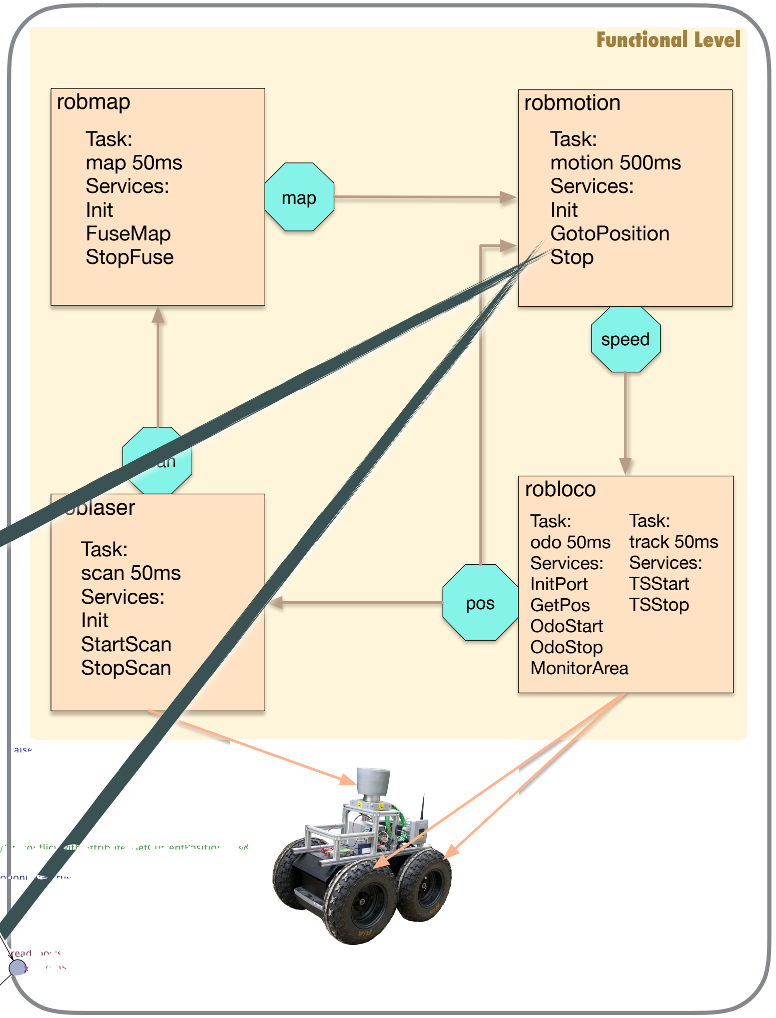


GenoM to UPPAAL

A template that produces the UPPAAL model of *any* GenoM specification for the PocoLibs implementation

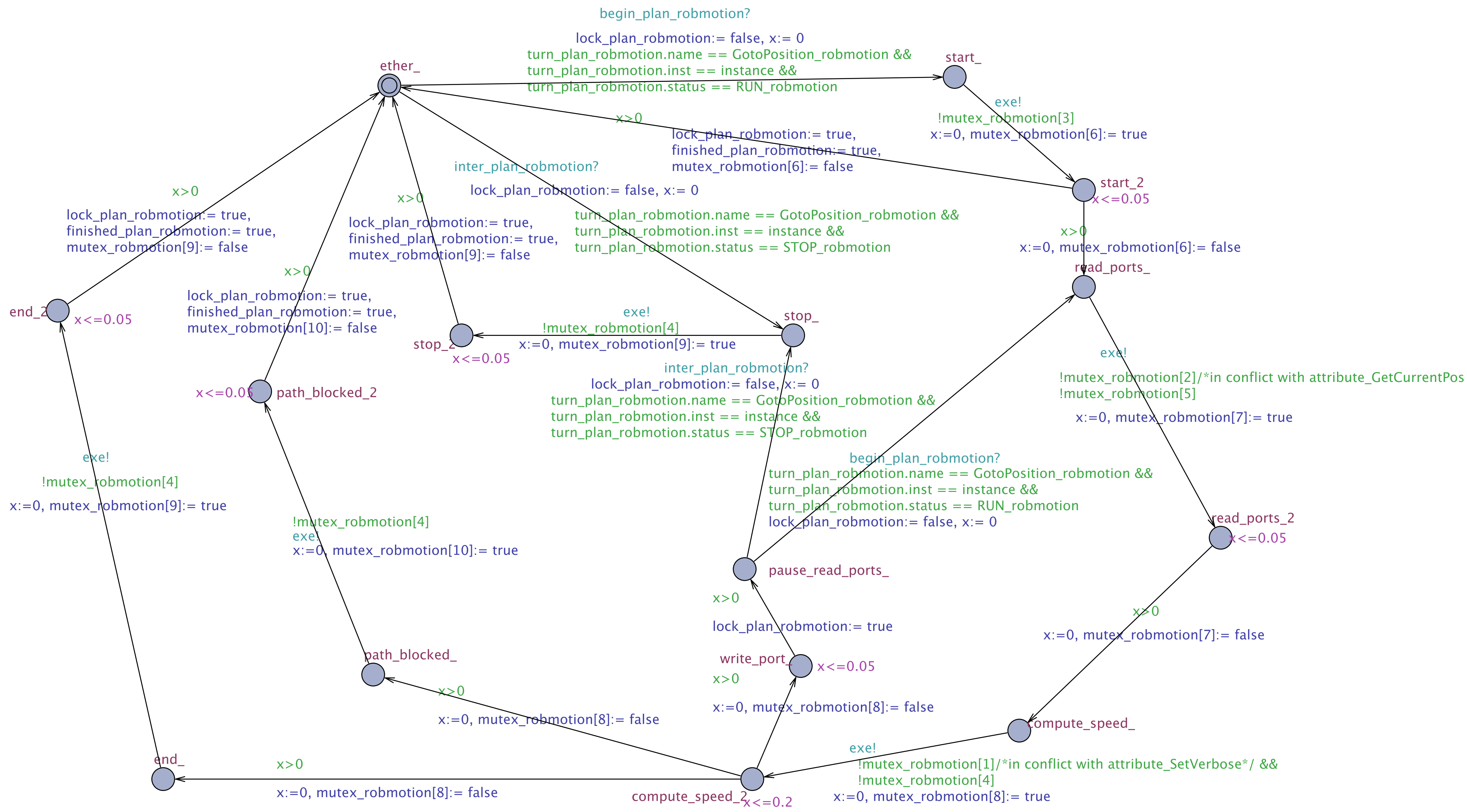
example:

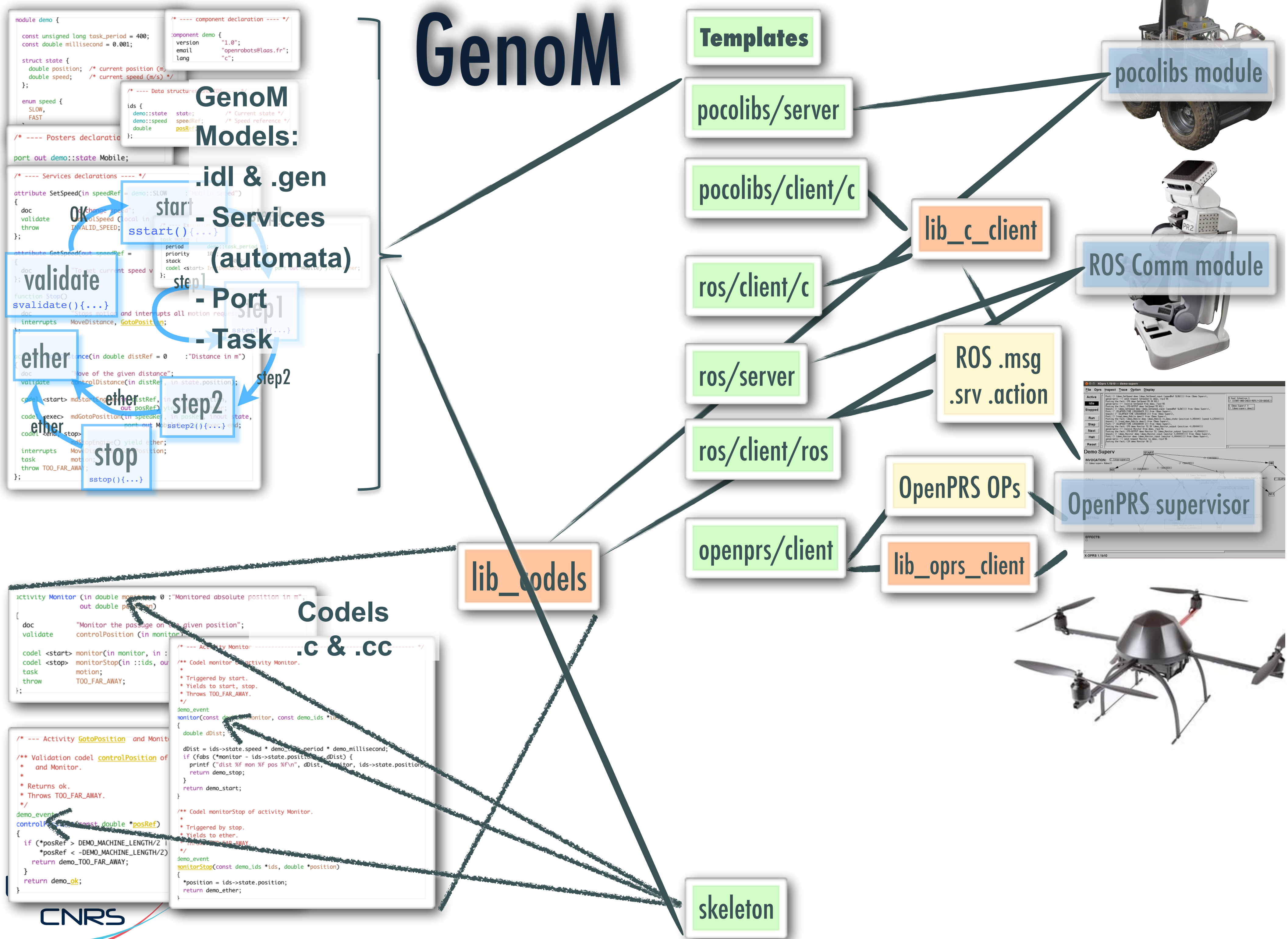


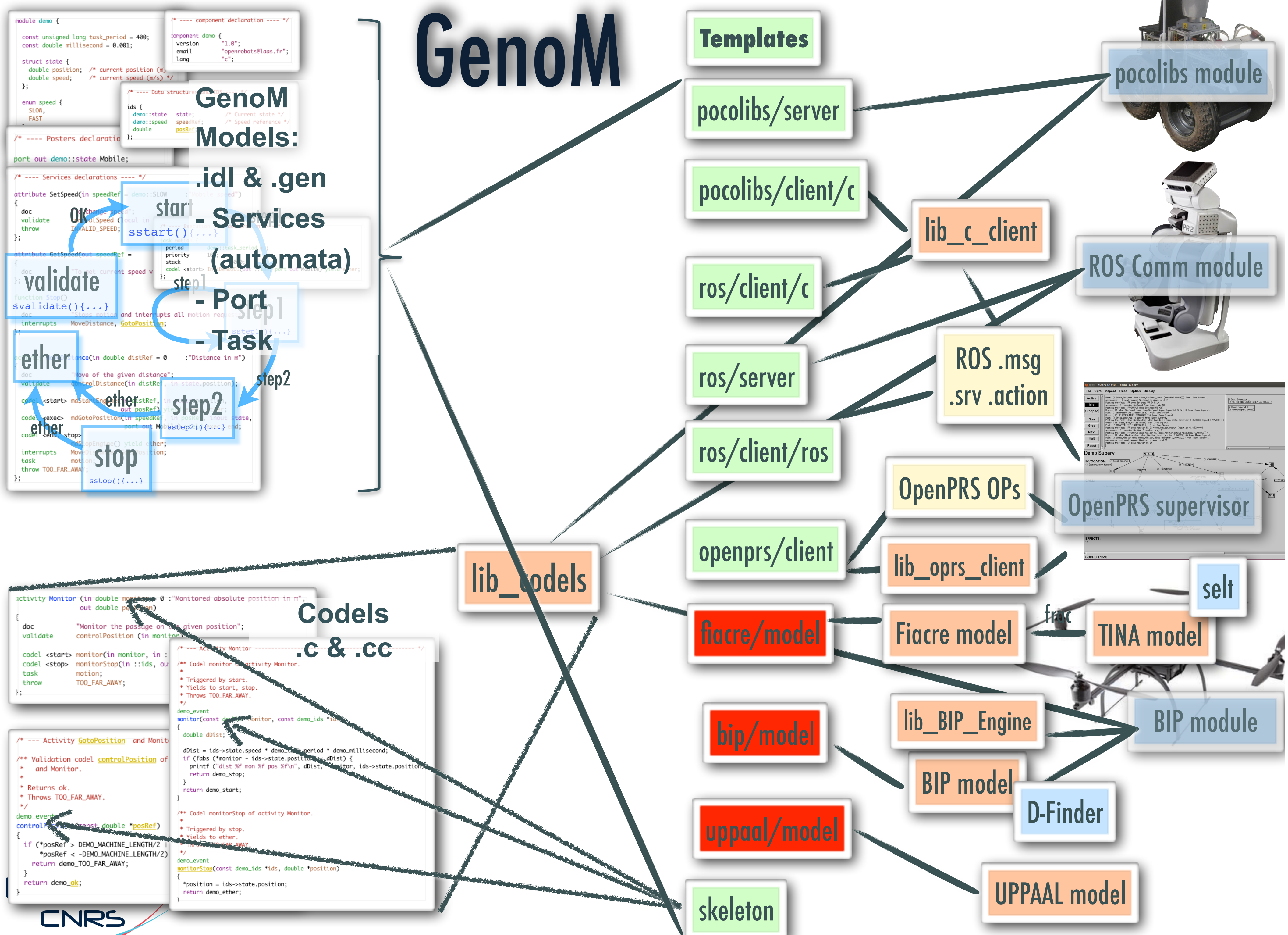


Model Checking

unspawned
end_spawn_plan_robmotion?
receive
!sched_robmotion
recv_robmotion?
sched_robmotion:= false
sched_robmotion
recv_urg_robmotion?
!shut_robmotion
launch_serv_robmotion(tab_robmotion)
exe!
GetExploredMap_2
GetExploredMap_ !mutex_robmotion[0] && !mutex_robmotion[7]
x<=0
x:=0, mutex_robmotion[5]:= true
req_robmotion==GetExploredMap_robmotion
SetVerbose_2
mutex_robmotion[5]:= false
exe!
x<=0
x:=0, mutex_robmotion[1]:= true
!mutex_robmotion[0]/*in conflict with InitPlanParameters_plan*/ && !mutex_robmotion[8]
SetVerbose
req_robmotion==SetVerbose_robmotion
mutex_robmotion[1]:= false
req_robmotion < 0
interrupt_cntrl_robmotion(req_robmotion,tab_robmotion,shut_robmotion)
finish
decode
req_robmotion==Stop_robmotion
x:= 0
Stop_
manage_intterupt_robmotion(req_robmotion, 0, tab_robmotion)
mutex_robmotion[2]:= false
GetCurrentPosition_2
x<=0
x<=0
shut_robmotion
req_robmotion==GetCurrentPosition_robmotion
!mutex_robmotion[0]/*in conflict with InitPlanParameters_plan*/ && !mutex_robmotion[7]
GetCurrentPosition_
x:=0, mutex_robmotion[2]:= true
mutex_robmotion[3]:= false
exe!
wait
GetGoalPosition_2
mutex_robmotion[4]:= false
exe!
x:=0, mutex_robmotion[3]:= true
x<=0
req_robmotion==GetGoalPosition_robmotion
!mutex_robmotion[0]/*in conflict with InitPlanParameters_plan*/ &&
!mutex_robmotion[6]
exe!
launch_serv_robmotion(tab_robmotion)
GetCurrentSpeed_2
off_signal_robmotion(tab_robmotion)
GetGoalPosition_
exe!
!mutex_robmotion[0] &&
x<=0
!mutex_robmotion[10] &&
req_robmotion==GetCurrentSpeed_robmotion
!mutex_robmotion[8] &&
shutdown
GetCurrentSpeed_
!mutex_robmotion[9]
x:= 0
x:=0, mutex_robmotion[4]:= true
x>0
slot_search_robmotion (req_robmotion, tab_robmotion)
req_robmotion==GotoPosition_robmotion
val_GotoPosition
x<=0.01
req_robmotion >=InitSpeedRefPort_robmotion &&
req_robmotion <=GotoPosition_robmotion &&
noval_robmotion(req_robmotion)
x>0
manage


begin_plan_robmotion?
lock_plan_robmotion:= false, x:= 0
turn_plan_robmotion.name == GotoPosition_robmotion &&
turn_plan_robmotion.inst == instance &&
turn_plan_robmotion.status == RUN_robmotion
ether_
start_
exe!
!mutex_robmotion[3]
x:=0, mutex_robmotion[6]:= true
x>0
lock_plan_robmotion:= true,
finished_plan_robmotion:= true,
mutex_robmotion[6]:= false
inter_plan_robmotion?
lock_plan_robmotion:= false, x:= 0
turn_plan_robmotion.name == GotoPosition_robmotion &&
turn_plan_robmotion.inst == instance &&
turn_plan_robmotion.status == STOP_robmotion
start_2
x<=0.05
x>0
x:=0, mutex_robmotion[6]:= false
read_ports_
x>0
lock_plan_robmotion:= true,
finished_plan_robmotion:= true,
mutex_robmotion[9]:= false
x>0
lock_plan_robmotion:= true,
finished_plan_robmotion:= true,
mutex_robmotion[9]:= false
x>0
lock_plan_robmotion:= true,
finished_plan_robmotion:= true,
mutex_robmotion[10]:= false
end_2
x<=0.05
exe!
!mutex_robmotion[4]
x:=0, mutex_robmotion[9]:= true
stop_2
x<=0.05
stop_
inter_plan_robmotion?
lock_plan_robmotion:= false, x:= 0
turn_plan_robmotion.name == GotoPosition_robmotion &&
turn_plan_robmotion.inst == instance &&
turn_plan_robmotion.status == STOP_robmotion
x<=0.05
path_blocked_2
exe!
!mutex_robmotion[2]/*in conflict with attribute_GetCurrentPos
!mutex_robmotion[5]
x:=0, mutex_robmotion[7]:= true
exe!
!mutex_robmotion[4]
x:=0, mutex_robmotion[9]:= true
begin_plan_robmotion?
turn_plan_robmotion.name == GotoPosition_robmotion &&
turn_plan_robmotion.inst == instance &&
turn_plan_robmotion.status == RUN_robmotion
lock_plan_robmotion:= false, x:= 0
!mutex_robmotion[4]
exe!
x:=0, mutex_robmotion[10]:= true
read_ports_2
x<=0.05
pause_read_ports_
x>0
lock_plan_robmotion:= true
x>0
x:=0, mutex_robmotion[7]:= false
path_blocked_
write_port_
x<=0.05
x>0
x:=0, mutex_robmotion[8]:= false
x>0
x:=0, mutex_robmotion[8]:= false
compute_speed_
end_
x>0
x:=0, mutex_robmotion[8]:= false
exe!
!mutex_robmotion[1]/*in conflict with attribute_SetVerbose*/ &&
!mutex_robmotion[4]
x:=0, mutex_robmotion[8]:= true
compute_speed_2
x<=0.2


GenoM
GenoM Models:
.idl & .gen
- Services (automata)
- Port
- Task
start
validate
ether
step1
step2
stop
OK
sstart(){...}
svalidate(){...}
sstep2(){...}
sstop(){...}
Codels
.c & .cc
Templates
pocolibs/server
pocolibs/client/c
ros/client/c
ros/server
ros/client/ros
openprs/client
skeleton
lib_codels
lib_c_client
ROS .msg .srv .action
OpenPRS OPs
lib_oprs_client
pocolibs module
ROS Comm module
OpenPRS supervisor
CNRS


GenoM
GenoM Models:
.idl & .gen
- Services (automata)
- Port
- Task
start
validate
ether
step1
step2
stop
Codels
.c & .cc
Templates
pocolibs/server
pocolibs/client/c
ros/client/c
ros/server
ros/client/ros
openprs/client
fiacre/model
bip/model
uppaal/model
skeleton
lib_codels
lib_c_client
ROS .msg .srv .action
OpenPRS OPs
lib_oprs_client
Fiacre model
lib_BIP_Engine
BIP model
UPPAAL model
pocolibs module
ROS Comm module
OpenPRS supervisor
TINA model
selt
frac
BIP module
D-Finder
CNRS

Verification results FIACRE/TINA

✓ Schedulability of execution tasks

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

Verification with TINA: FALSE

✓ Progress of activities

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

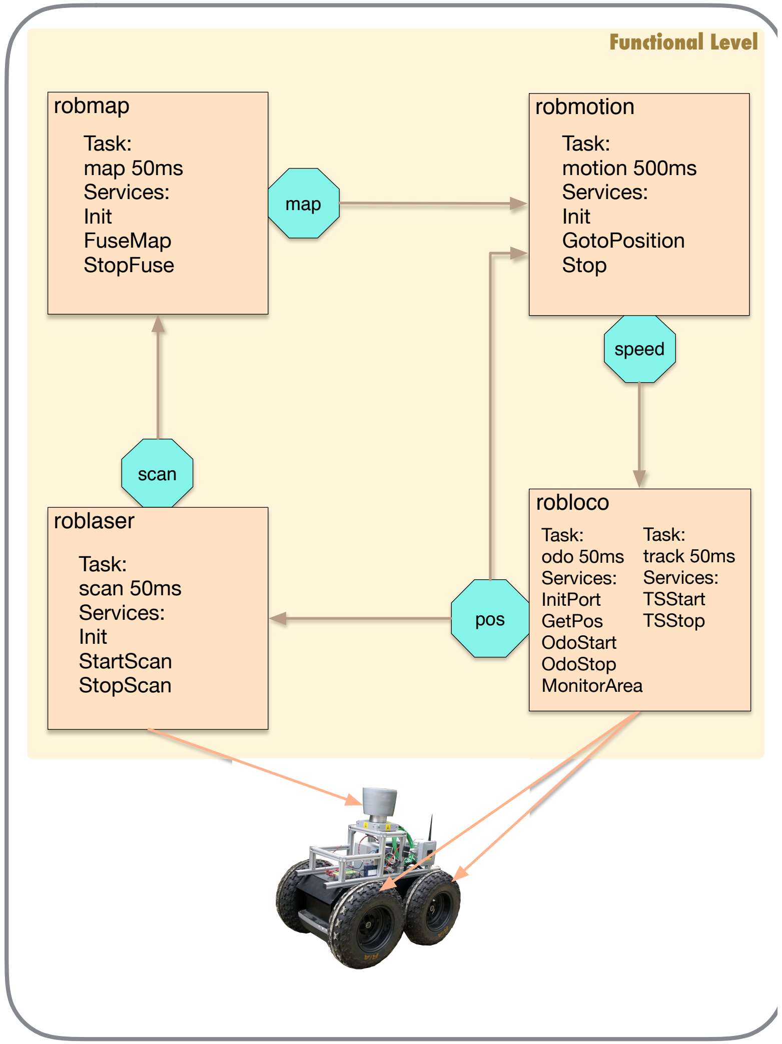
Verification with TINA: TRUE

✓ Position port update bounded in time

Final result 1 second and 274 ms

✓ RobMotion Stop leads to RobLoco stopping the robot

property bounded_stop_1 is (robmotion/control_task/state Stop_req) leadsto (robmotion/GotoPosition/state stop) within [0,0.5]
property bounded_stop_2 is (robmotion/GotoPosition/state stop) leads to leave (robloco/TSStart/state update) within [0,0.06]



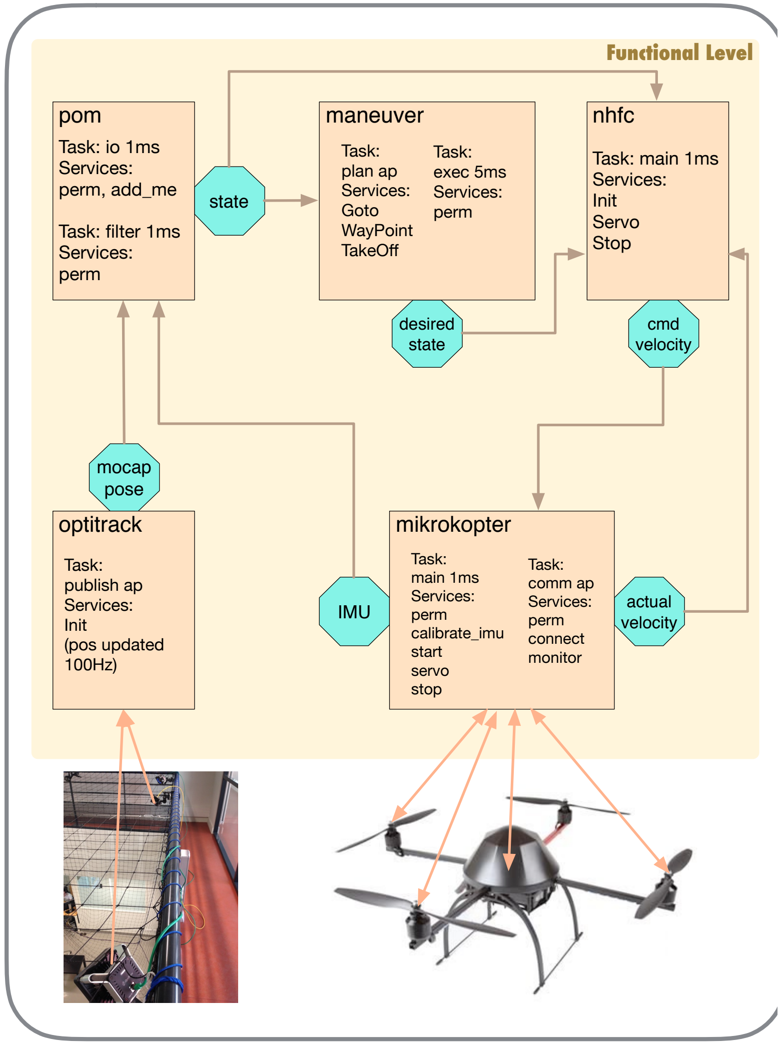
Verification results FIACRE/TINA

✓ Schedulability of execution tasks

property schedulability_main is always (microkopter/main/state executing ⇒ not (main_period_signal))

Verification with TINA: FALSE

Hold for all tasks with an octo-core but not with a quad-core ODROID-C0



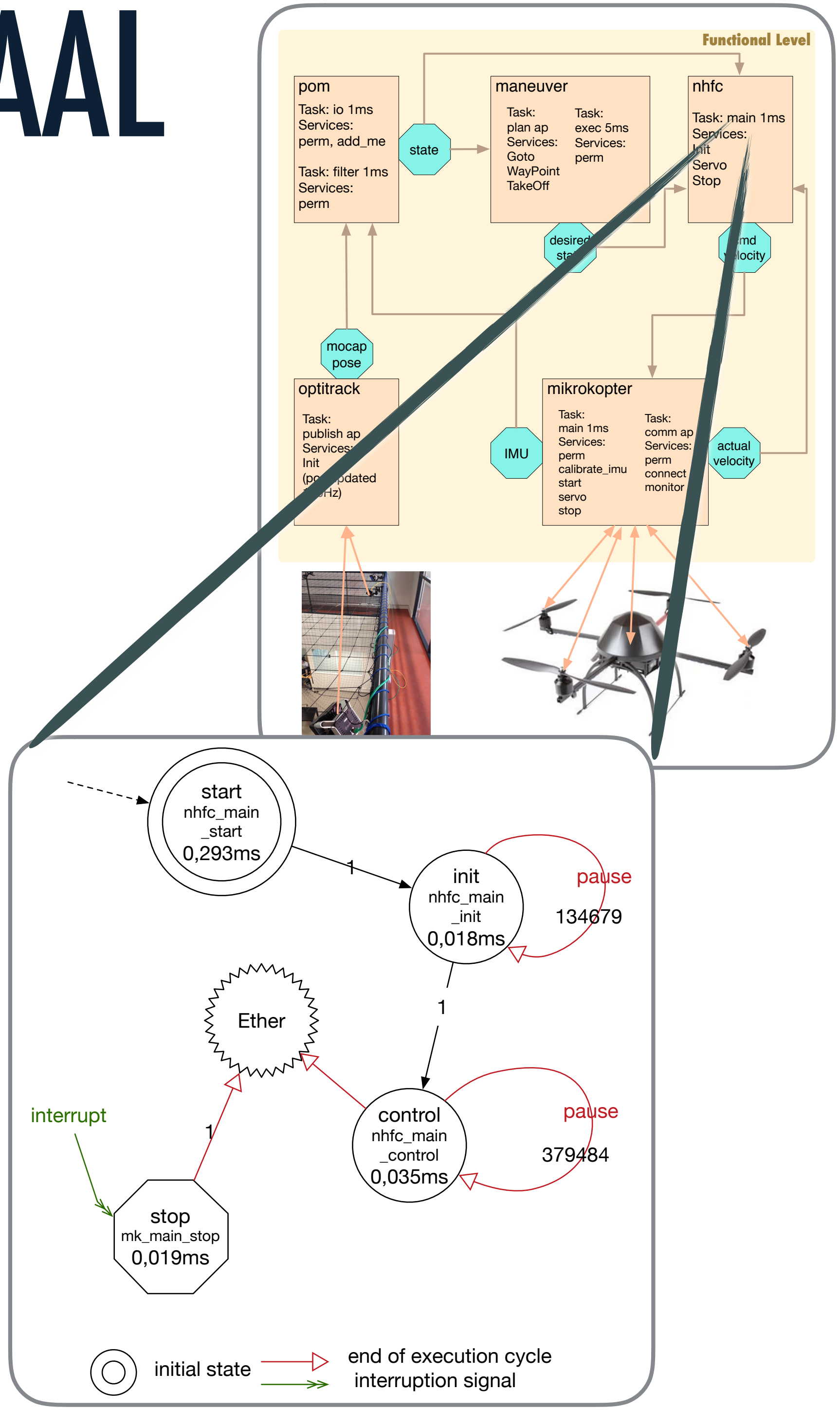
Verification with UPPAAL

- ✓ Overall, similar properties than the one expressed in Fiacre
- ✓ SMC extension to take into account the probability transition in the service automata

```
codel<start> nhfc_main_start(…) yield init;
codel<init> nhfc_main_init(…)yield pause::init, control;
codel<control> nhfc_main_control(…)yield pause::control;
codel<stop> mk_main_stop(…)yield ether;
```

```
nhfc: 1 transitions for main, from nhfc_start to nhfc_init.
nhfc: 134679 transitions for main, from nhfc_init to nhfc_pause_init.
nhfc: 1 transitions for main, from nhfc_init to nhfc_control.
nhfc: 379484 transitions for main, from nhfc_control to nhfc_pause_control.
nhfc: 1 transitions for main, from nhfc_stop to nhfc_ether.
```

```
nhfc: nhfc_main_start called: 1 times, wcet: 0.000293.
nhfc: nhfc_main_init called: 134680 times, wcet: 0.000018.
nhfc: nhfc_main_control called: 379484 times, wcet: 0.000035.
nhfc: mk_main_stop called: 1 times, wcet: 0.000019.
```



Current GenoM V&V templates

Middleware / Framework

	Offline PocoLibs	Online PocoLibs	Online ROS
BIP	+ RT D-Finder	++	Under Dev
FIACRE	++	Under Dev	Proposal
UPPAAL	+++		
UPPAAL SMC	++		

- The Fiacre-PocoLibs template is complete and tested on numerous modules (model over multiple modules and ports communication), UPPAAL has a slight performance advantage.
- The BIP-PocoLibs model is complete, but has been a disappointment with respect to RT D-Finder
- The BIP-PocoLibs model for the BIP Engine is complete and functional, but the BIP Engine needs more work

Current state, limit of the approach

- Added the **WCET** declaration in the .gen specification file, but we are **NOT** checking codels
- We are not checking against a model of the environment
- Specific scheduling policy (no preemption) and codel non-interruptibility...
- Still requires good knowledge of GenoM **AND** the formal framework as to write properties to check, and analyse the results...

Conclusion

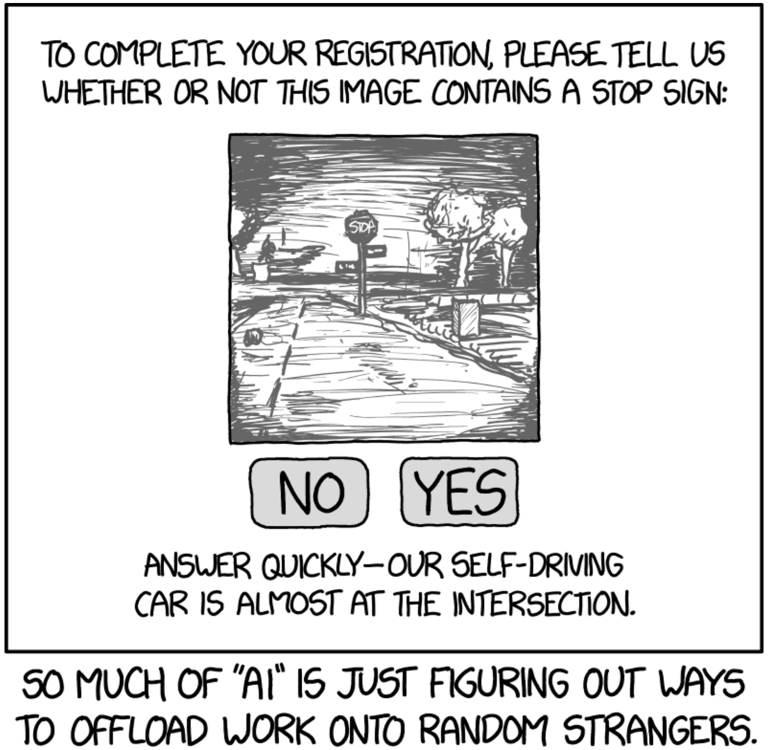
- We derive a formal model from robotic functional component specification
- We get a very fine grained and low level formal model of the complete functional layer internal code execution and interactions
- Three V&V techniques are considered: model checking (TINA/Fiacre & UPPAAL) and automatic invariant composition and satisfiability (BIP/RT-D-Finder).
- BIP and now Fiacre also provide an engine to run the model

LAAS CNRS

Long term research agenda

- Run Time Verification
- Deeper model (codel arguments, SDI, algo, check the codel, etc)
- Evolve toward decisional level (Planning/Acting/Monitoring)
- Clarify platform dependent model (scheduling policy, #CPU/#Core)

Thanks to



Verimag: Saddek Bensalem, Jacques Combaz, Souha Ben-Rayana

LAAS/VerTICS: Bernard Berthomieu, Silvano Dal Zilio, Pierre Emanuel Hladik

Mälardalen University: Cristina Seceleanu

Part of this work is funded by the
H2020 European project CPSE Labs

LAAS CNRS