Validation and Verification of Autonomous Systems

Félix Ingrand
LAAS-RIS

Journée Perspectives et Défis de l'IA,
Véhicule Autonome et IA
Paris
October 11, 2018

Autonomous Vehicle Software








Autonomous Vehicle Software



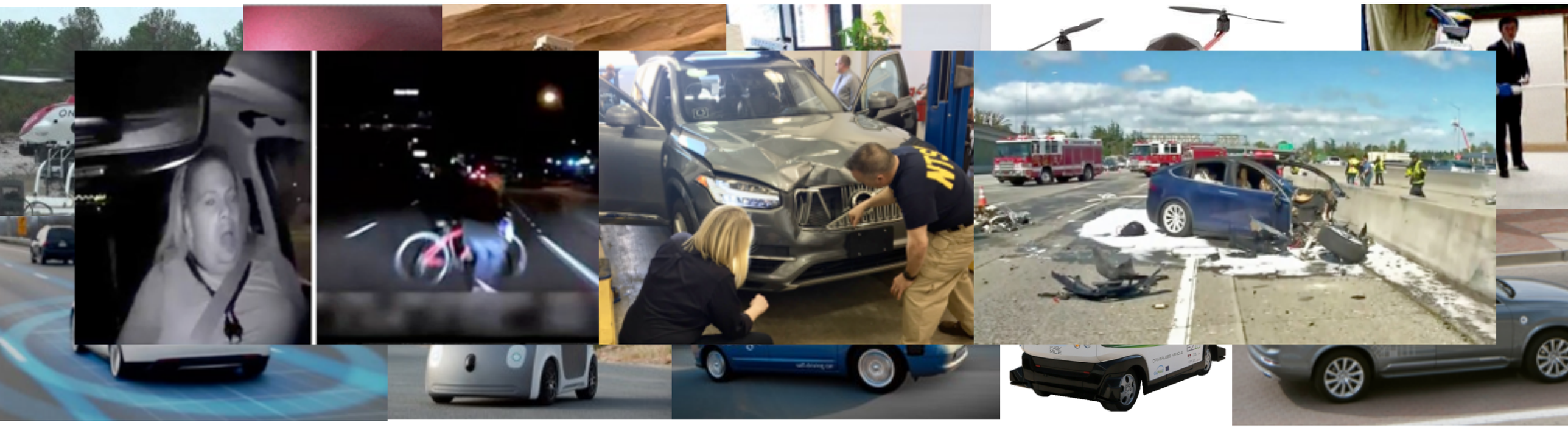





Software represents a large part of the development of Autonomous Vehicle, yet, most of it is not V&V...

LAAS CNRS

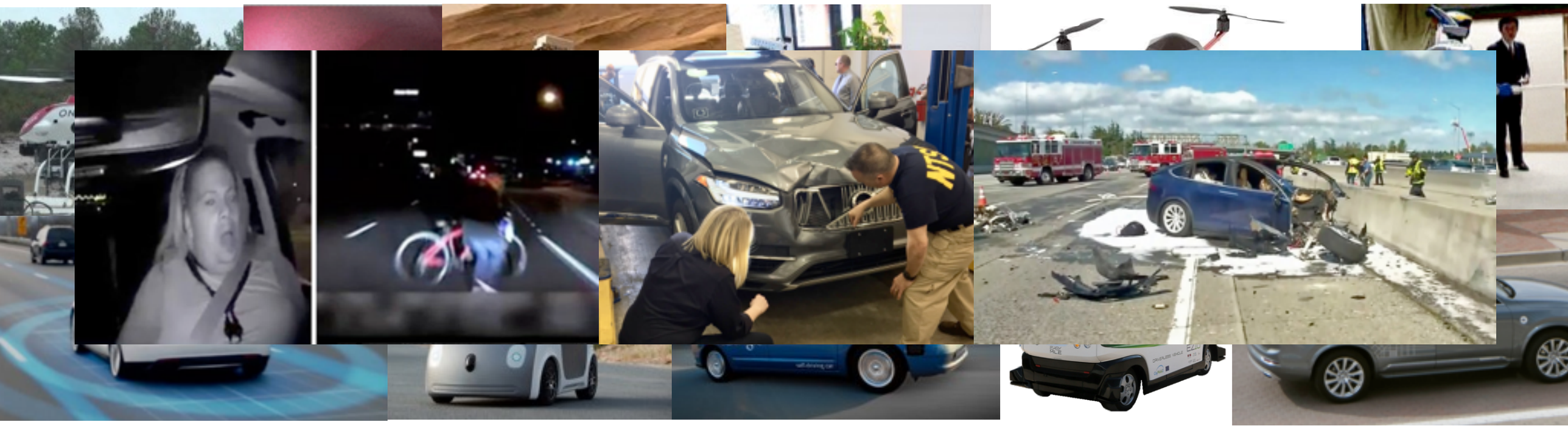
Autonomous Vehicle Software



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

... with dramatic consequences...

Autonomous Vehicle Software



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

... with dramatic consequences...

...while V&V is used for some of these complex (but not quite autonomous) systems

LAAS CNRS

Software Validation and Verification

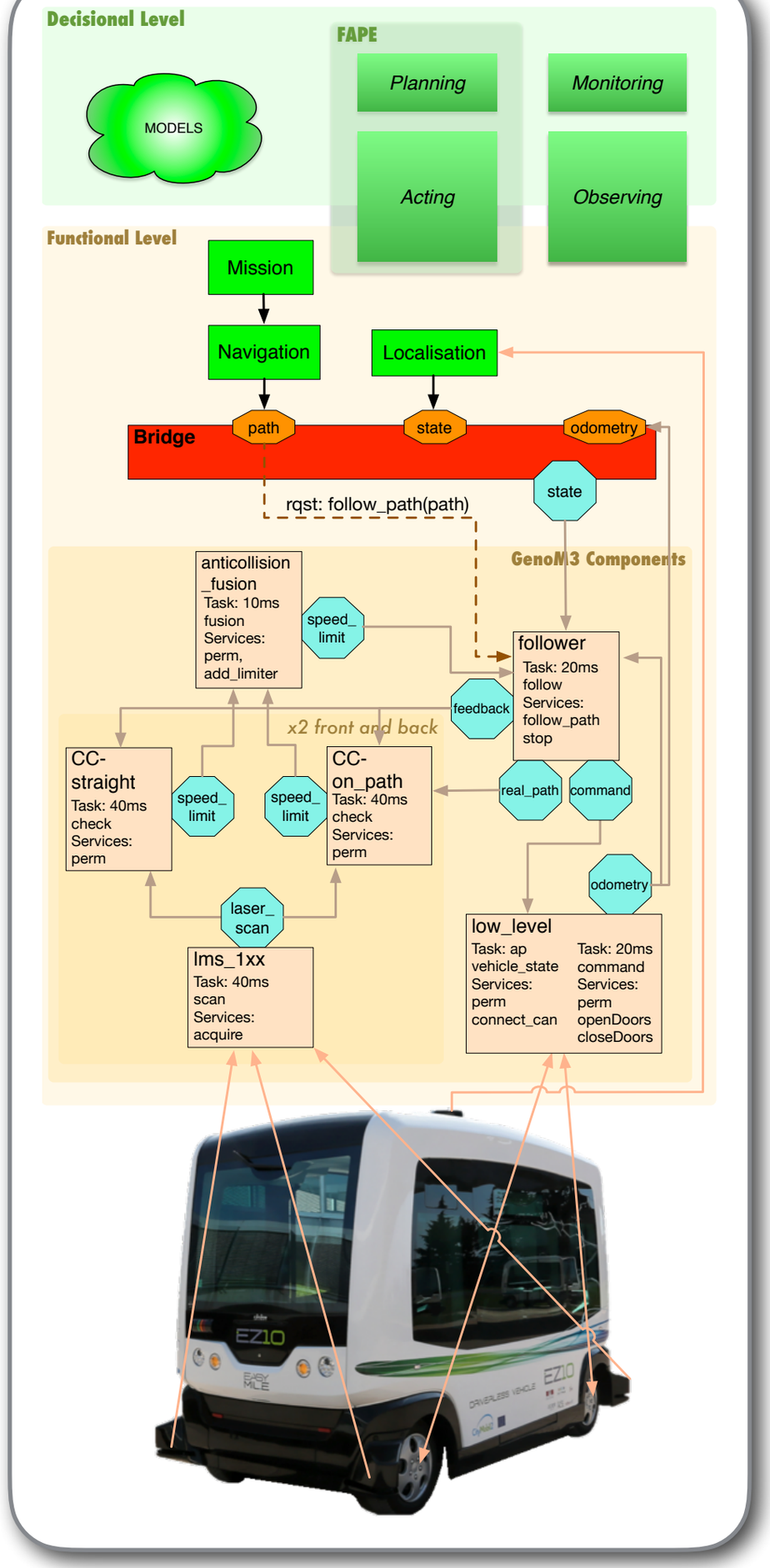
- Require formal models and mathematically/logically sound "checking" techniques
  - formal models (e.g., FSM, IO automata, Petri nets, timed automata, situation calculus, synchronous systems, etc)
  - checking by reachable state exploration (e.g. model checking), logical induction (e.g. theorem proving, sat solving, etc) or runtime verification
  - complete methods, over approximation, statistical methods, etc...

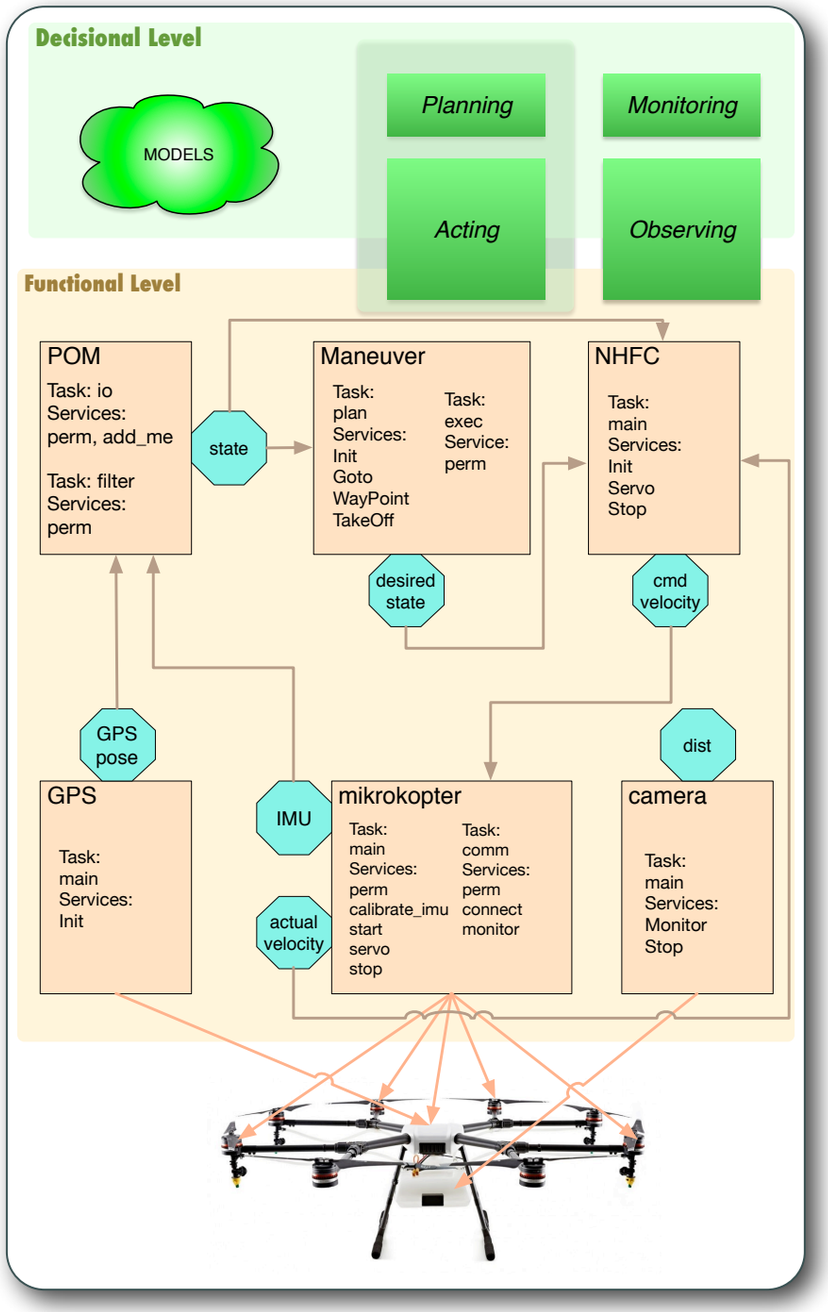
LAAS CNRS

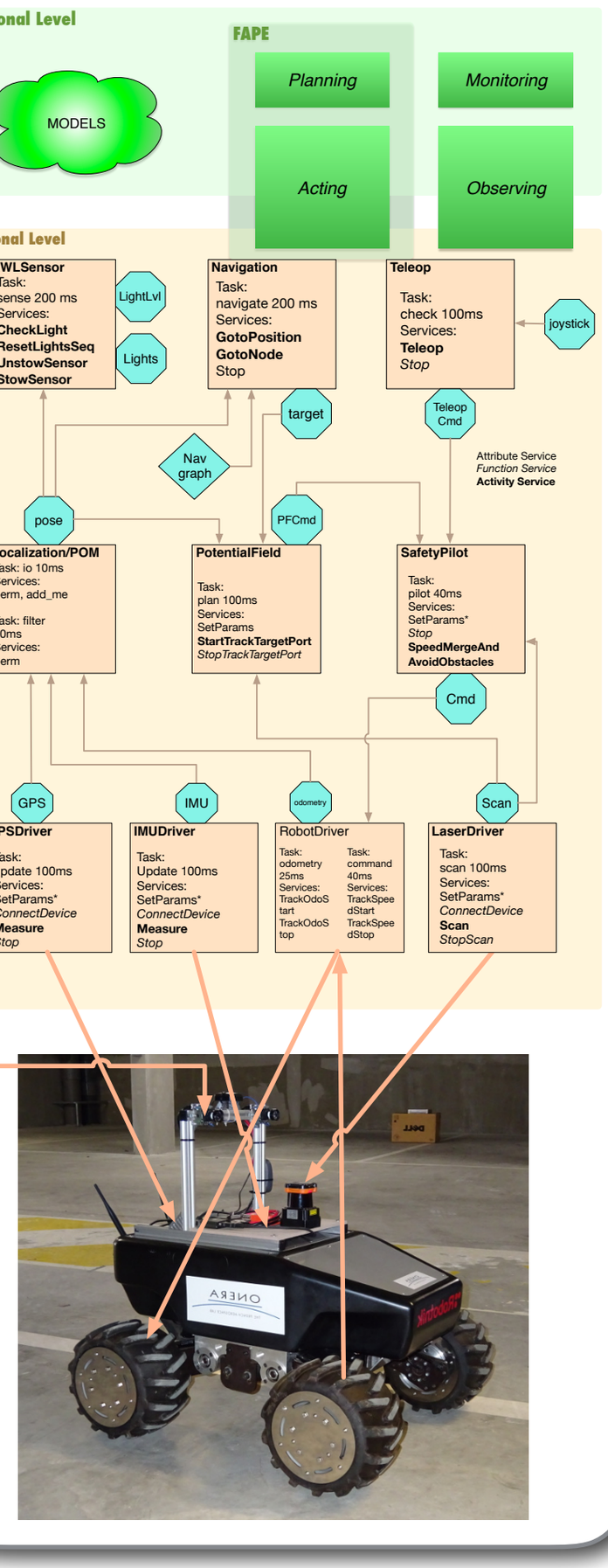
V&V on Robotic Software

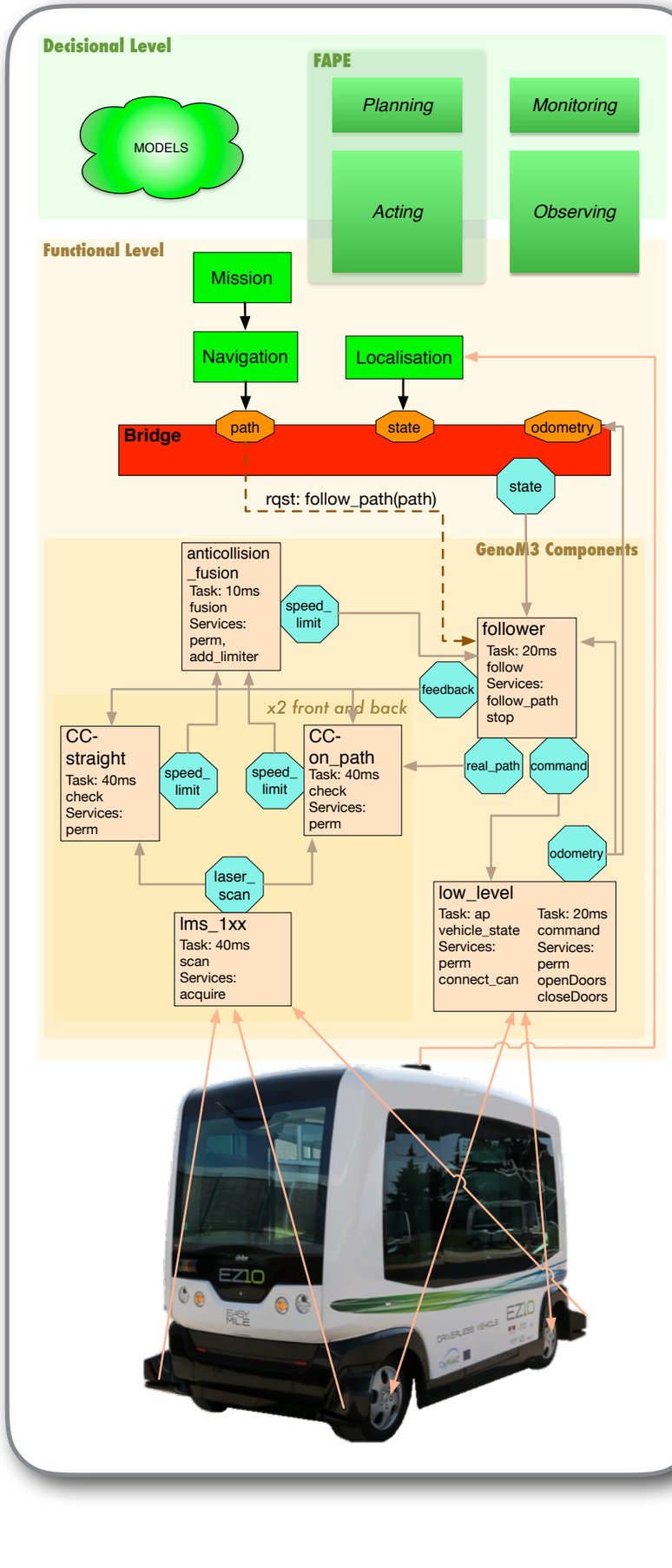
Check that the autonomous shuttle drives safely e.g.:

- Plan is safe and executable
- Stop in time when an obstacle has been detected
- The door does no open while moving
- Path following remains in bound
- Check that the vehicle has a consistant perception/ action loop
- Speed command is produced “timely”
- Laser scan – freq and range
- Speed control – freq and value
- Time for an emergency stop









V&V models: Different situations over a complete autonomous system

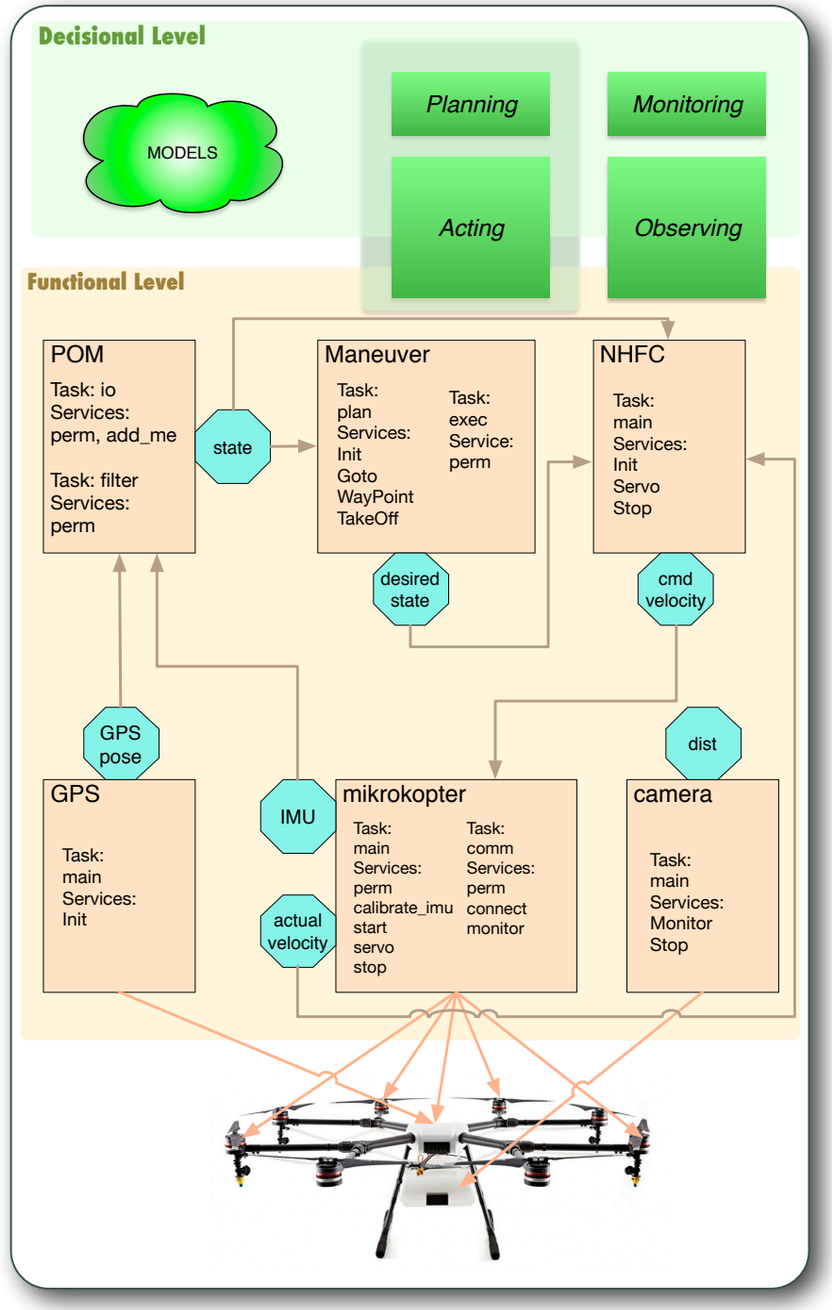
Formal models: decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR, observing

Learned models: Reinforcement learning models, perception models, etc.

Specification models: Software engineering models: e.g. GenoM3, Oroccos, MAUVE, RobotML, etc.

Programming directly the Model: Orccad, Scade, etc.

No Model...



V&V models: Different situations over a complete autonomous system

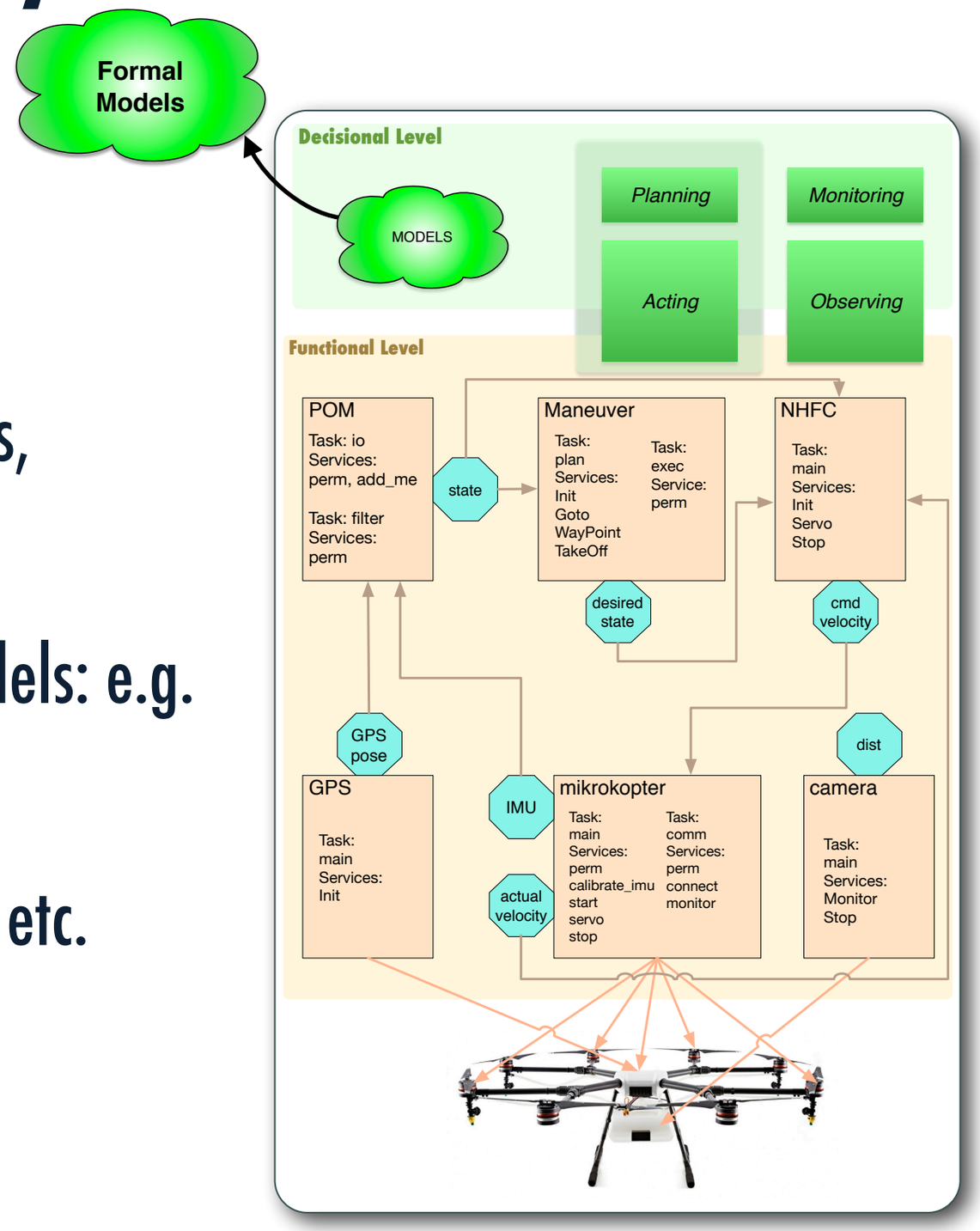
Formal models: decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR, observing

Learned models: Reinforcement learning models, perception models, etc.

Specification models: Software engineering models: e.g. GenoM3, Oroccos, MAUVE, RobotML, etc.

Programming directly the Model: Orccad, Scade, etc.

No Model...



V&V models: Different situations over a complete autonomous system

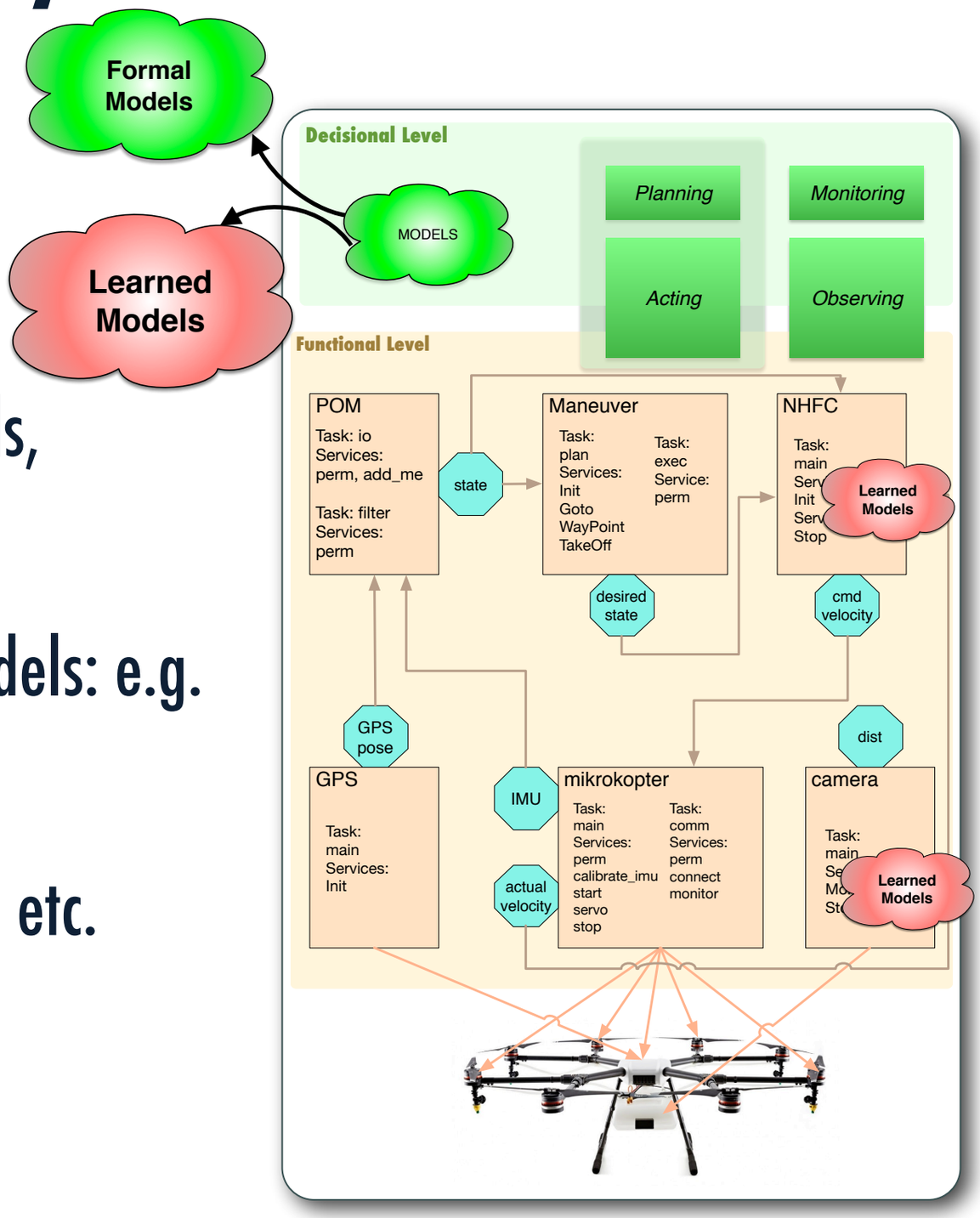
Formal models: decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR, observing

Learned models: Reinforcement learning models, perception models, etc.

Specification models: Software engineering models: e.g. GenoM3, Oroccos, MAUVE, RobotML, etc.

Programming directly the Model: Orccad, Scade, etc.

No Model...



V&V models: Different situations over a complete autonomous system

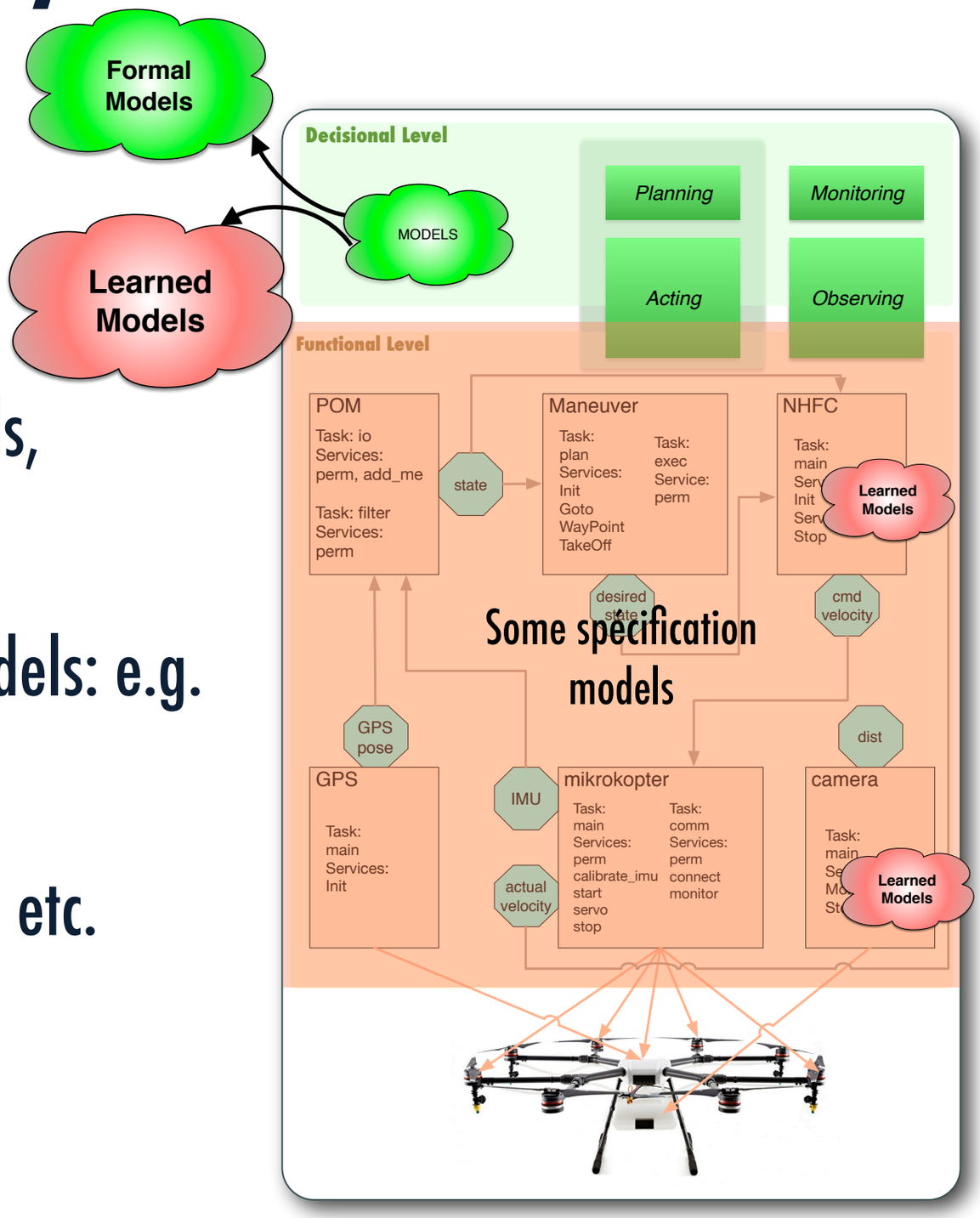
Formal models: decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR, observing

Learned models: Reinforcement learning models, perception models, etc.

Specification models: Software engineering models: e.g. GenoM3, Oroccos, MAUVE, RobotML, etc.

Programming directly the Model: Orccad, Scade, etc.

No Model...



V&V models: Different situations over a complete autonomous system

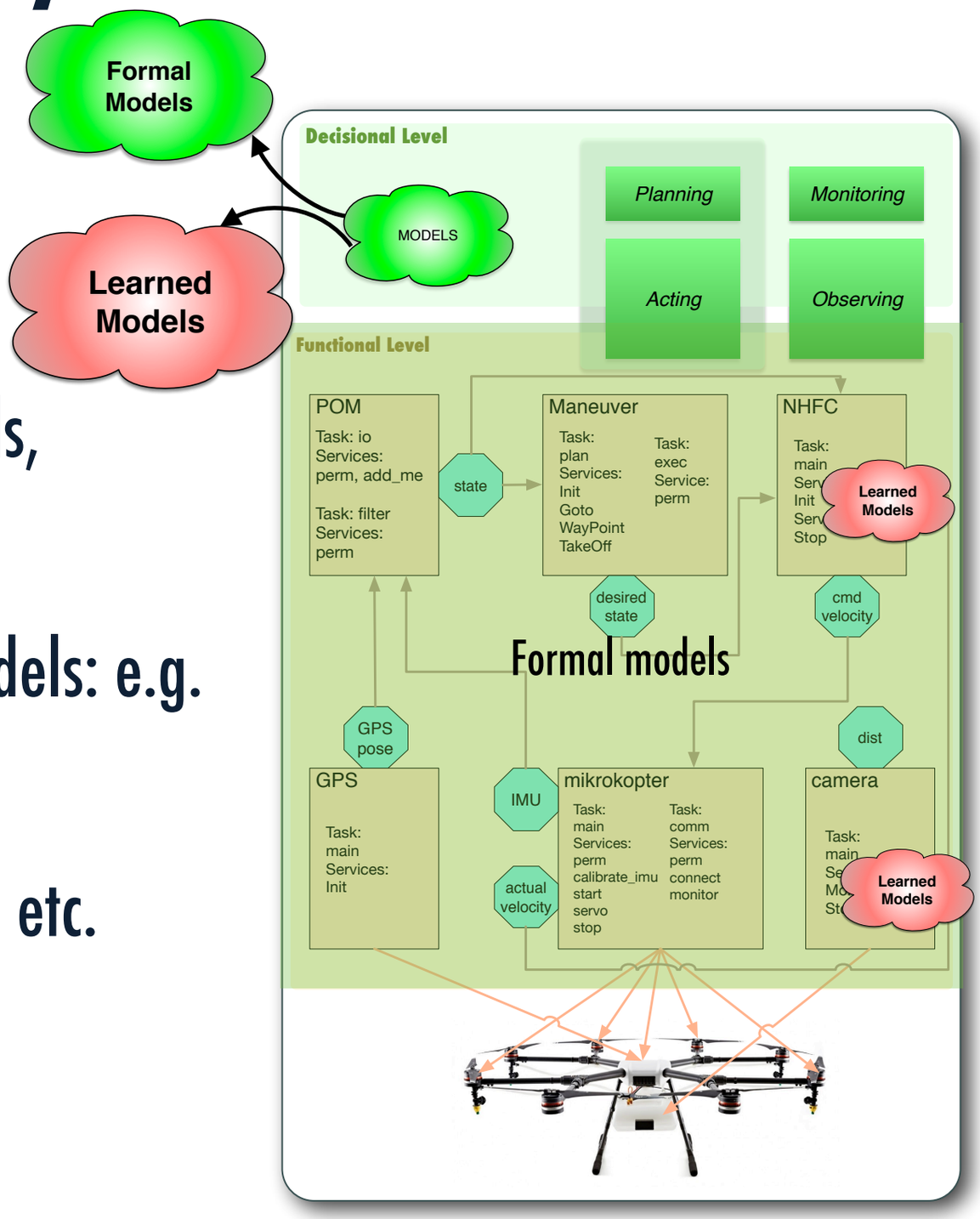
Formal models: decisional : planning (e.g. UPPAAL, model checking), monitoring, FDIR, observing

Learned models: Reinforcement learning models, perception models, etc.

Specification models: Software engineering models: e.g. GenoM3, Oroccos, MAUVE, RobotML, etc.

Programming directly the Model: Orccad, Scade, etc.

No Model...



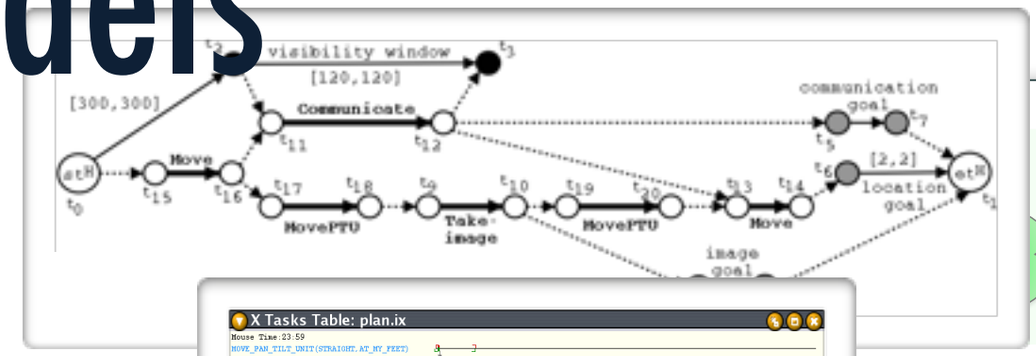
Decisional models

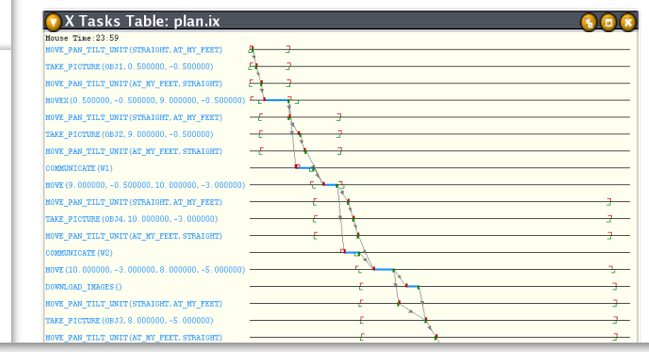
Planning

- Temporal Planning
  - TALPlanner [3], IxTeT, IDEA/T-ReX [2] (CSP and STN)
- Model checking
  - State reachability from the initial situation
  - UPPAAL-TiGA
    - LAAS from IxTeT action model, (planning and execution) [1]
    - CNR together with APSI (plan checking and execution)

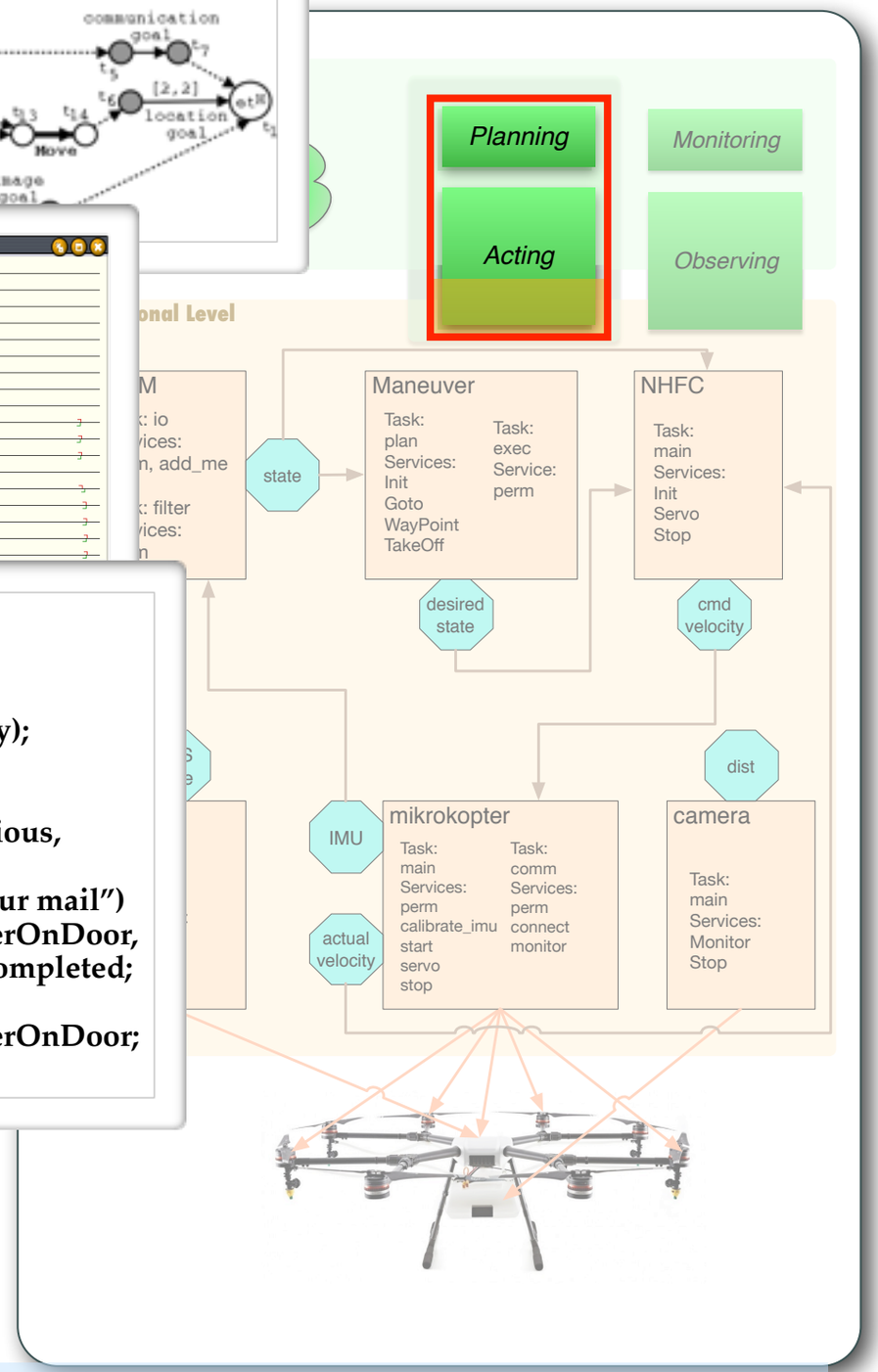
Acting

- TDL / Livingstone and SMV [4] (Model Checking)
- Situation Calculus Planning/Acting formalism [5]






```
Goal deliverMail (int room)
{
  double x, y;
  getRoomCoordinates(room, &x, &y);
  spawn navigateToLocn(x, y);
  spawn centerOnDoor(x, y)
      with sequential execution previous,
        terminate in 0:0:30.0;
  spawn speak("Xavier here with your mail")
      with sequential execution centerOnDoor,
        terminate at monitorPickup completed;
  spawn monitorPickup()
      with sequential execution centerOnDoor;
}
```



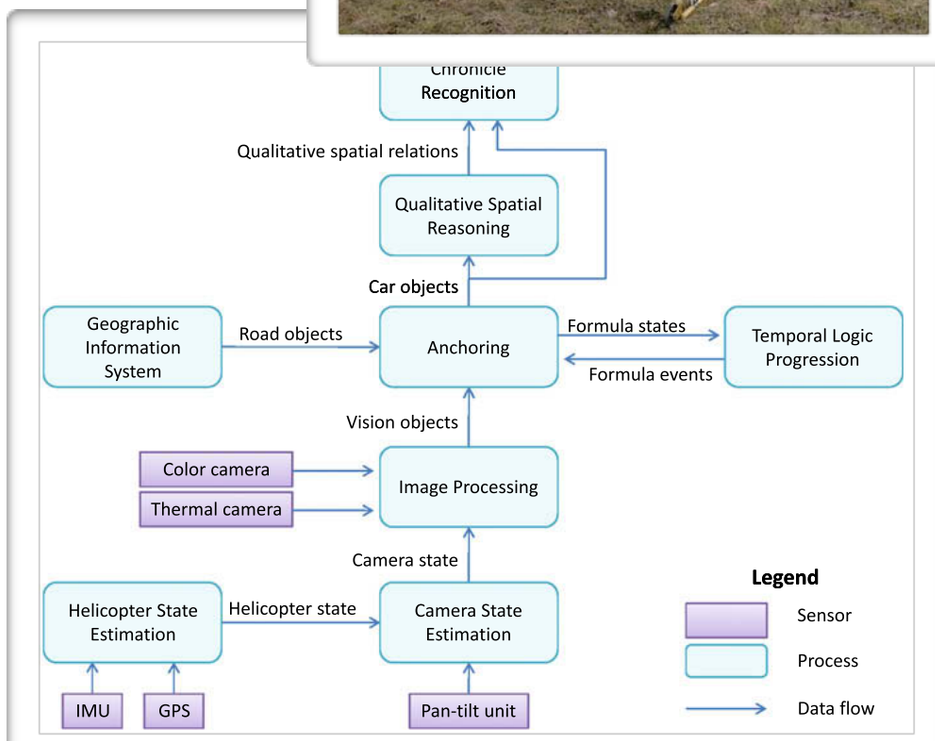
[1] Y. Abdeddaim, E. Asarin, M. Gallien, F. Ingrand, C. Lesire, and M. Sighireanu, "Planning Robust Temporal Plans , A Comparison Between CBTP and TGA Approaches," International Conference on Automated Planning and Scheduling, 2007, no. Providence, RI.
[2] F. Py, K. Rajan, and C. McGann, "A Systematic Agent Framework for Situated Autonomous Systems," International Conference on Autonomous Agents and Multiagent Systems, 2010.
[3] P. Doherty and J. Kvarnstram, "TALplanner: A temporal logic-based planner," AI Magazine, vol. 22, no. 3, p. 95, 2001.
[4] R. Simmons and C. Pecheur, "Automating Model Checking for Autonomous Systems," presented at the AAAI Spring Symposium on Real-Time Autonomous Systems, 2000.
[5] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," National Conference on Artificial Intelligence, 1998, pp. 11–18.

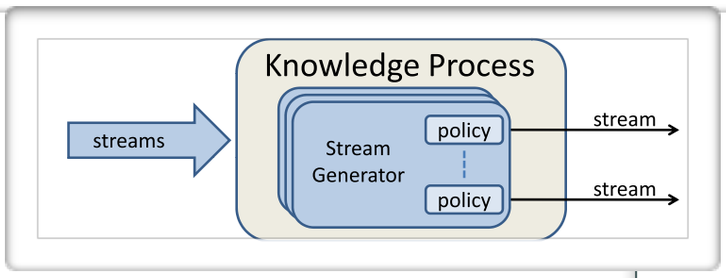
LAAS CNRS

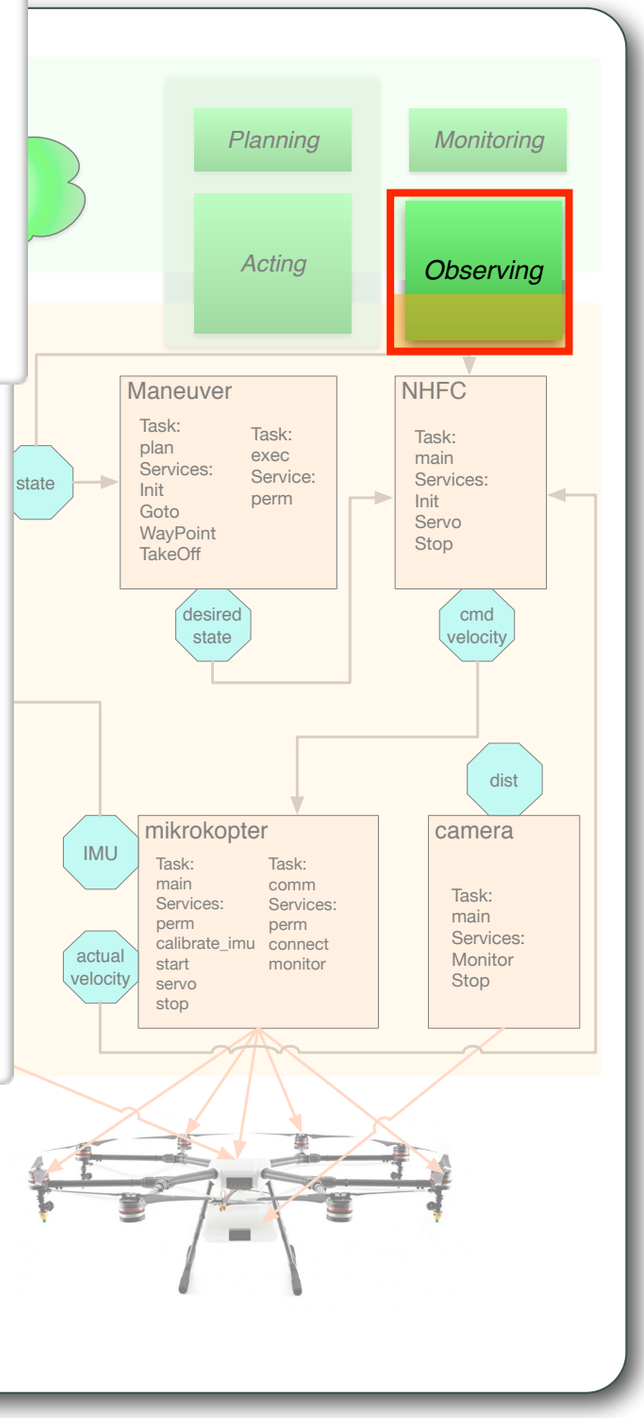
Decisional models

Observing: DyKnow [1]

- Comprehensive and coherent approach for observing
- They build on a stream based formalism on process:
  - primitive, refinement, configuration, mediation processes
  - policies over processes (temporal constraints)
- Data flow architecture (somewhat orthogonal to control flow architecture like GenoM)
- Still an interesting formalism which potentially opens a large field for V&V





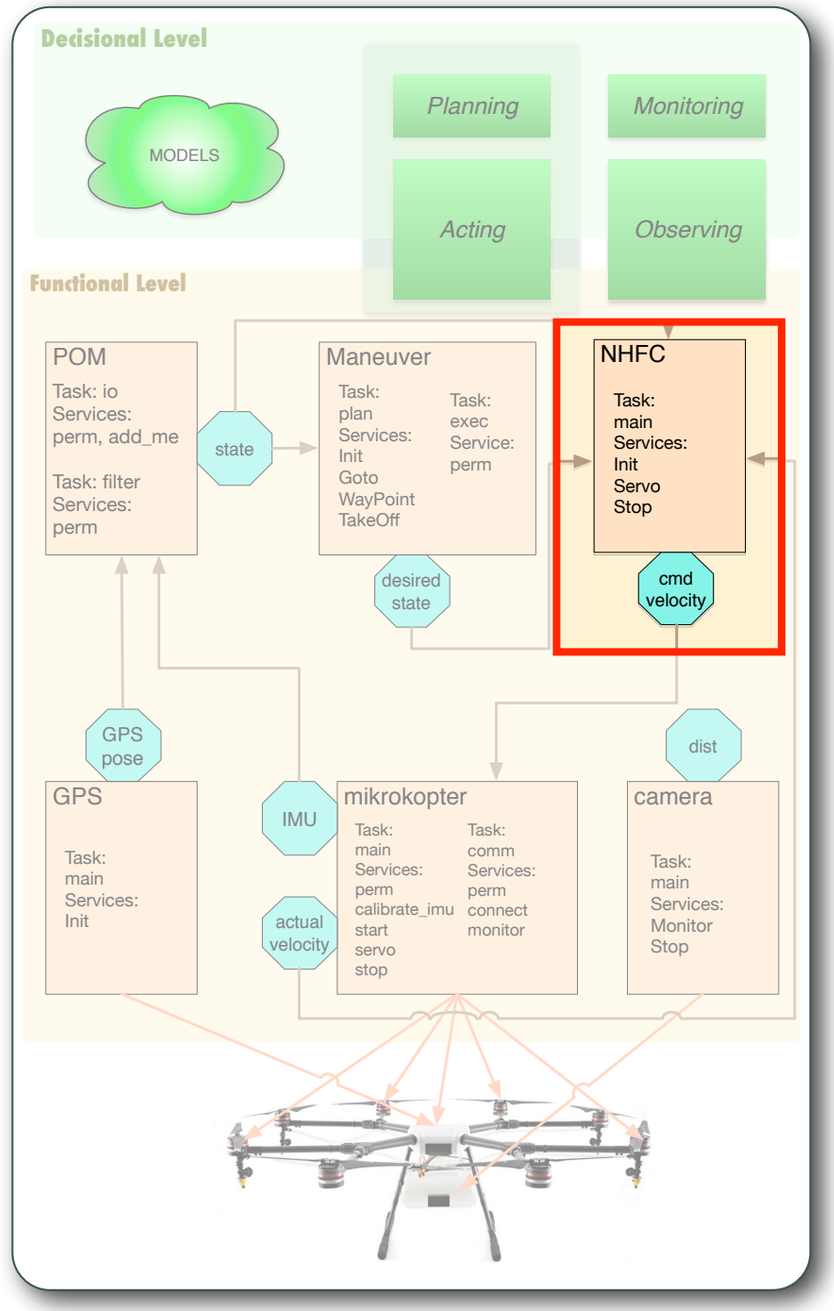


[1] F. Heintz, J. Kvarnström, and P. Doherty, "Bridging the sense-reasoning gap: DyKnow-Stream-based middleware for knowledge processing," Advanced Engineering Informatics, vol. 24, no. 1, pp. 14–26, 2010.

Directly using Formal Framework

- Synchronous approach
  - Orccad (Esterel) [1,2]
    - Control flow
    - SCADE/Lustre
  - Data flow
  - Signal [3]
  - Clocks

```
do [
    DrillMoveTo ();
    CloseImagerMoveTo ();
    [
        CloseImagerMonitor() ||
        DrillExtractSample()
    ]
] watching Alarm do
```



[1] B. Espiau and K. Kapellos, "Formal verification in robotics: Why and how?," ROBOTICS RESEARCH- …, 1996.

[2] T. J. Koo, B. Sinopoli, A. Sangiovanni-Vincentelli, and S. Sastry, "A formal approach to reactive system design: unmanned aerial vehicle flight management system design example," Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on, pp. 522–527, 1999.

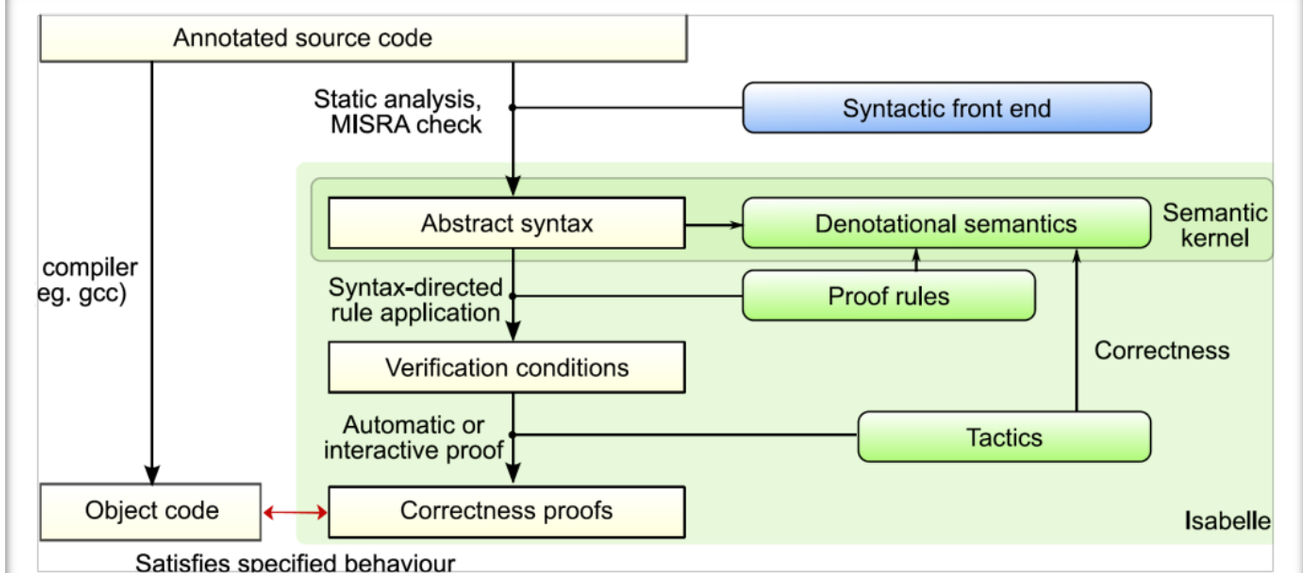
[3] E. Marchand, E. Rutten, H. Marchand, and F. Chaumette, "Specifying and verifying active vision-based robotic systems with the SIGNAL environment," International Journal of Robotics Research, vol. 17, no. 4, pp. 418–432, 1998.

No model… Still can extract or rebuild it

Robot navigation

- Code is written with correctness conditions in the code as pre- and post-conditions
- Very tedious (you have to annotate all the functions you want to prove)
- In [1] the authors show that this approach was accepted by the german certification authority IEC 61508 (SIL 3)

```
/*@
  @requires directions_len - 1 <= result_len_max && 0 <= rbuffer &&
            ${let V = ^Vector2DList{directions_data, directions_len}
               in (0 ∈ conv ^Vector2DSet{polygon_data, polygon_len}) ∧
                  (sorted-by (λv1 v2. angle v1 v2 ∈ {ϕ. 0 < ϕ ∧ ϕ < π}) V) ∧ (∀v ∈ set V. |v| = 1) }
           && \unrelated(polygon_data, polygon_len, result_data, result_len_max)
           && \unrelated(directions_data, directions_len, result_data, result_len_max)
  @modifies result_data[:directions_len-1]
  @ensures \result == sams_safe -->
    ${let E = ^RZList{result_data, directions_len - 1};
          P = ^Vector2DList{polygon_data, polygon_len};
          R = ^Vector2DList{directions_data, directions_len};
          SAFETYZONE = extend-by-radius `radius (convex-area P)
      in SAFETYZONE ∩ (angle-sector ^Vector2DR{&directions_data[0]}
                                    ^Vector2DR{&directions_data[directions_len-1]})
         ⊆ scan-field R E }
  @*/
SAMSStatus sampling( const Vector2D * polygon_data, Int32 polygon_len,
                     Float32 radius,
                     const Vector2D * directions_data, Int32 directions_len,
                     Int32 * result_data, Int32 result_len_max );
```

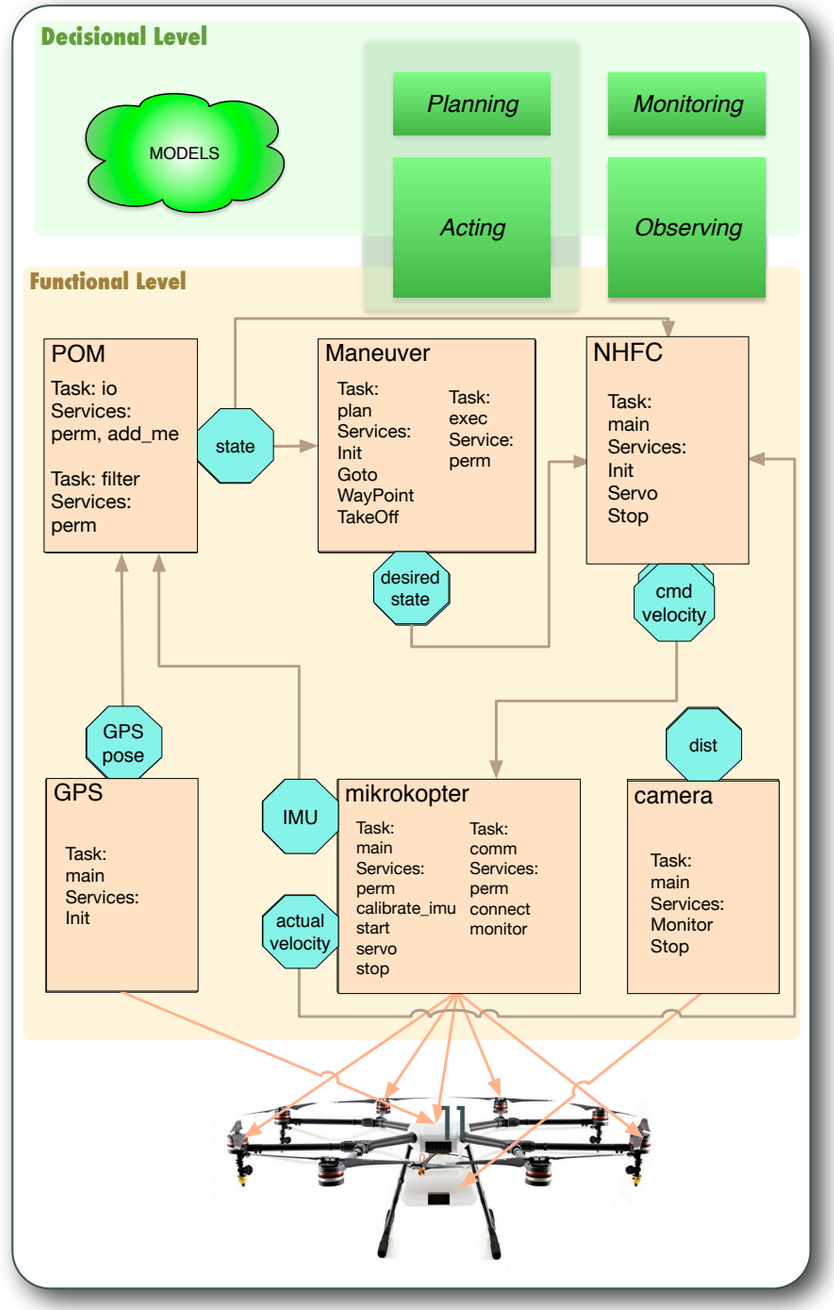


[1] H. Täubig, U. Frese, C. Hertzberg, C. Lüth, S. Mohr, E. Vorobev, and D. Walter, "Guaranteeing functional safety: design for provability and computer-aided verification," Auton Robot, vol. 32, no. 3, pp. 303–331, Dec. 2011.

From specification models to formal models

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

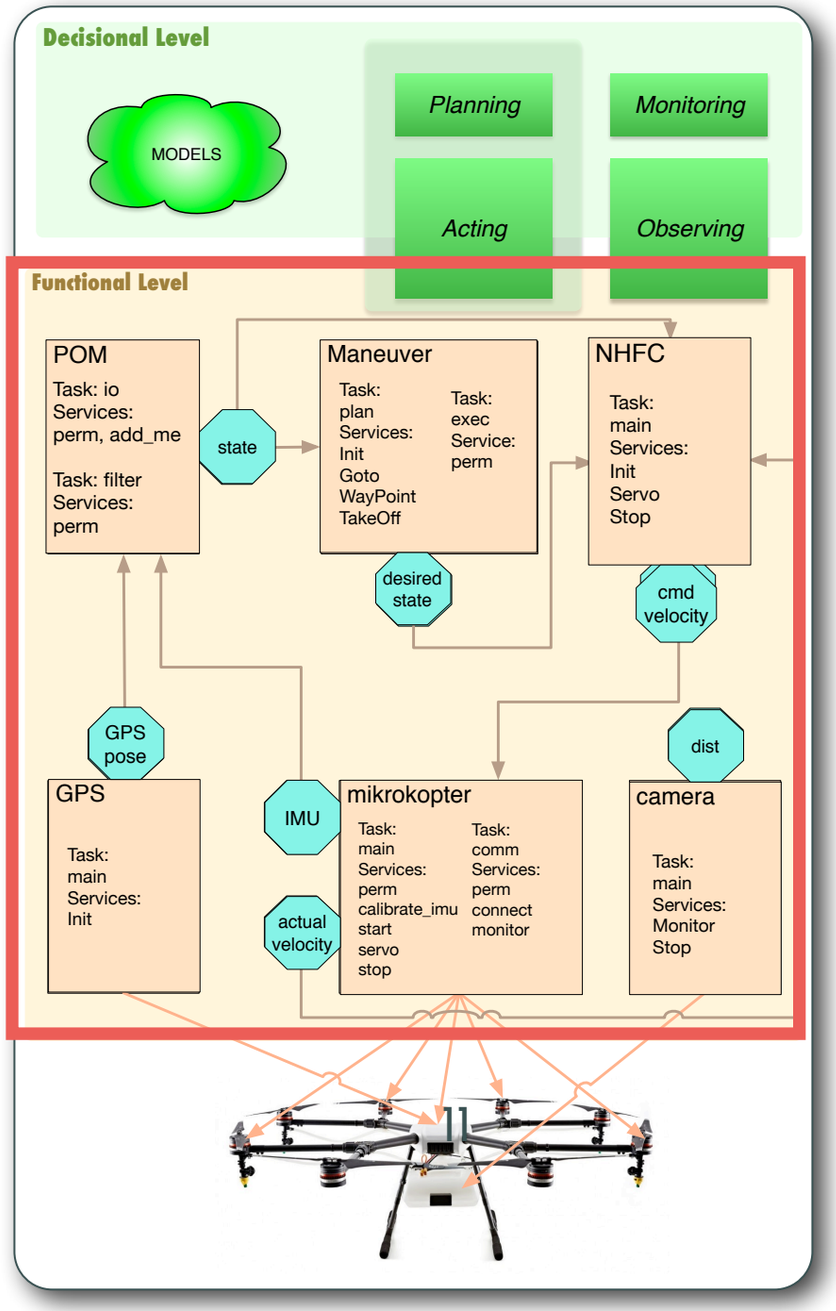
Specification: Model-Driven Software Engineering



From specification models to formal models

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

Specification: Model-Driven Software Engineering



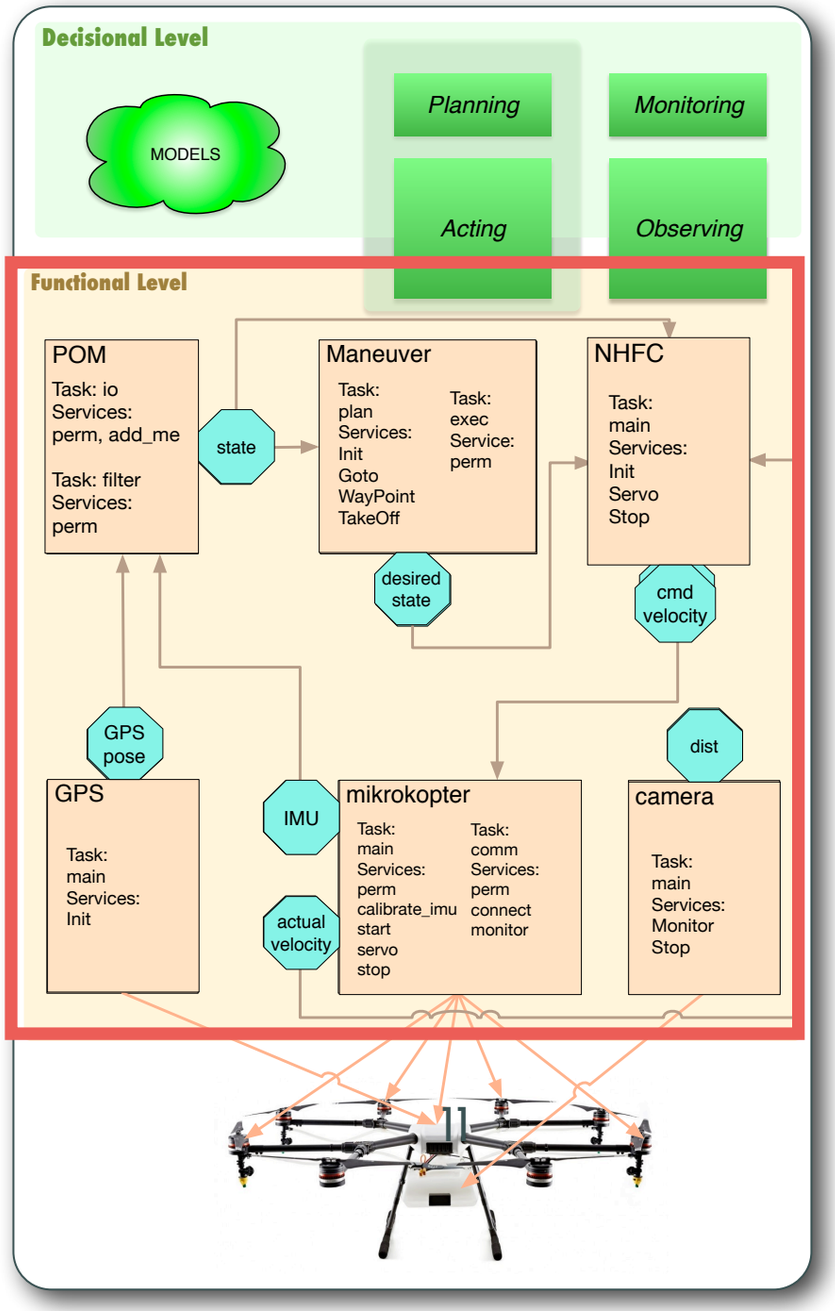
From specification models to formal models

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

Specification: Model-Driven Software Engineering

- BIP (Verimag)
- Fiacre/TINA (LAAS/VerTICS)
- UPPAAL (UPPsala & AALborg University)

Formal Methods/ Frameworks



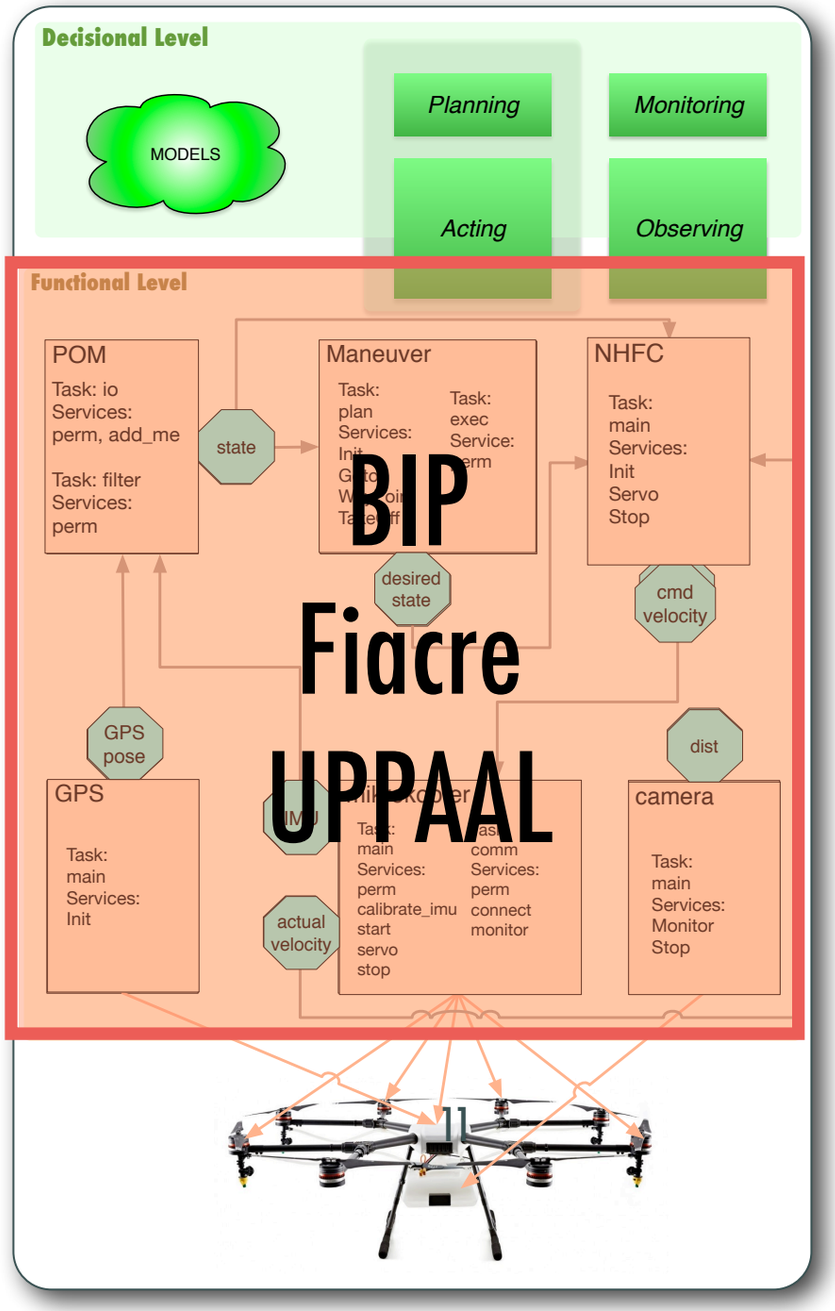
From specification models to formal models

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

Specification: Model-Driven Software Engineering

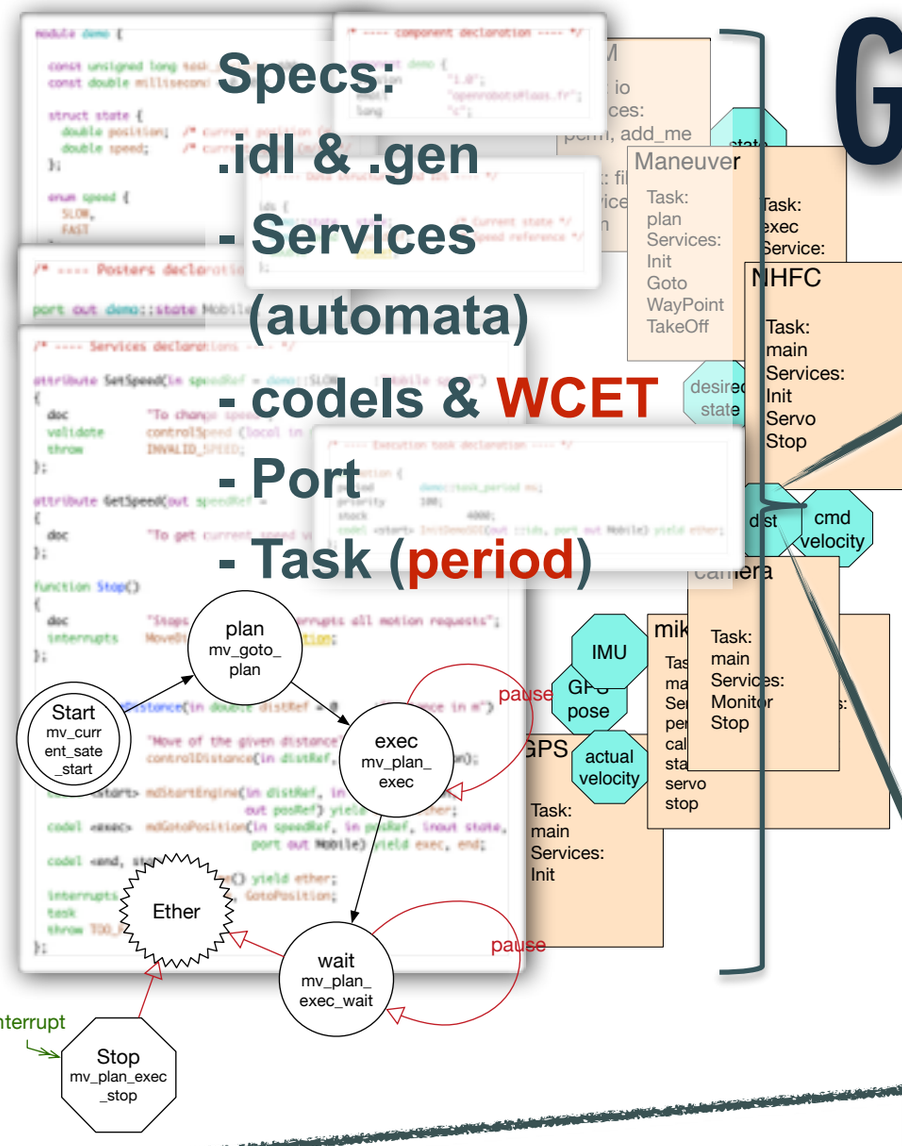
- BIP (Verimag)
- Fiacre/TINA (LAAS/VerTICS)
- UPPAAL (UPPsala & AALborg University)

Formal Methods/ Frameworks

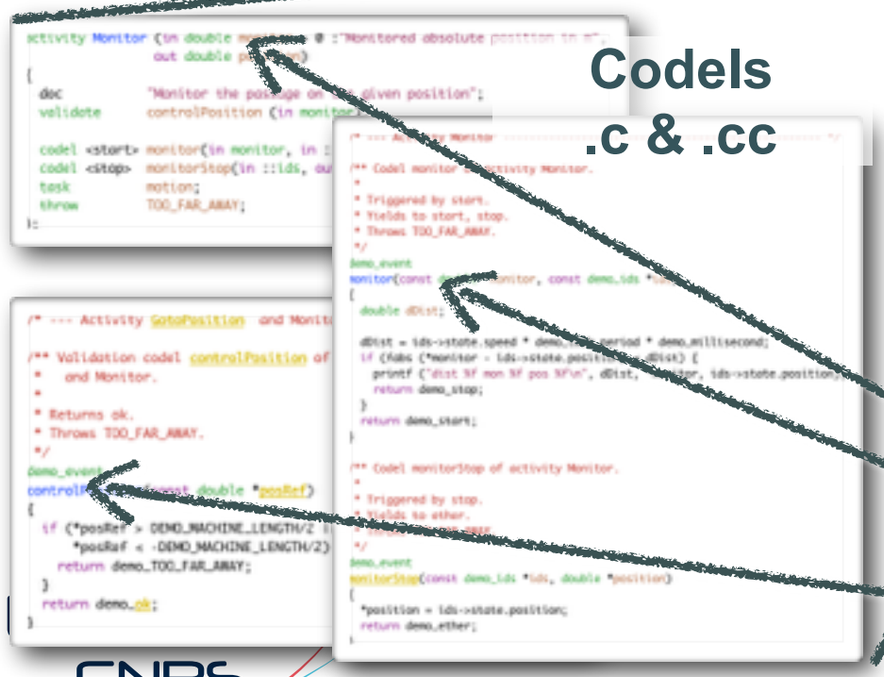


GenoM

Specs:
.idl & .gen
- Services
(automata)
- codels & WCET
- Port
- Task (period)



Codels
.c & .cc



Templates

pocolibs/server

pocolibs/client/c

lib_c_client

ros/client/c

ros/server

ros/client/ros

openprs/client

lib_oprs_client

lib_codels

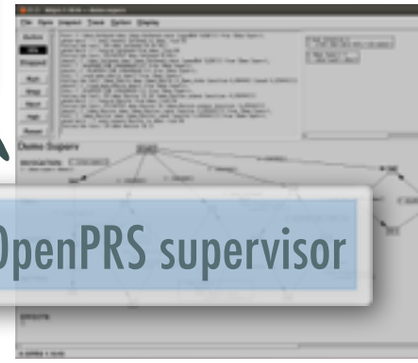
skeleton

pocolibs module

ROS Comm module

ROS .msg
.srv .action

OpenPRS OPs



OpenPRS supervisor

GenoM

Specs:
.idl & .gen
- Services
(automata)
- codels & WCET
- Port
- Task (period)

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

pocolibs module

ROS Comm module

ROS .msg
.srv .action

OpenPRS OPs

OpenPRS supervisor

lib_oprs_client

selt

frac

Fiacre model

TINA model

lib_BIP_Engine

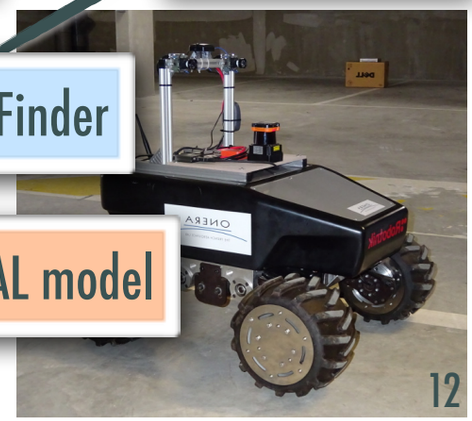
BIP module

BIP model

D-Finder

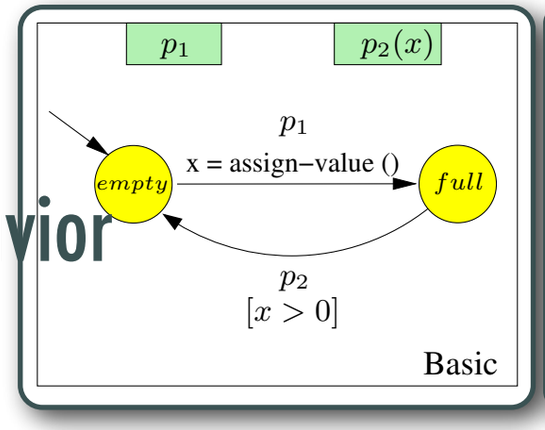
UPPAAL model

Codels
.c & .cc



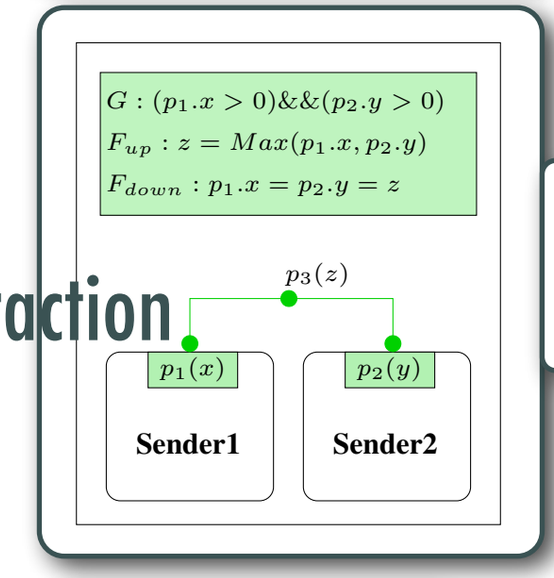
BIP Model example

Behavior



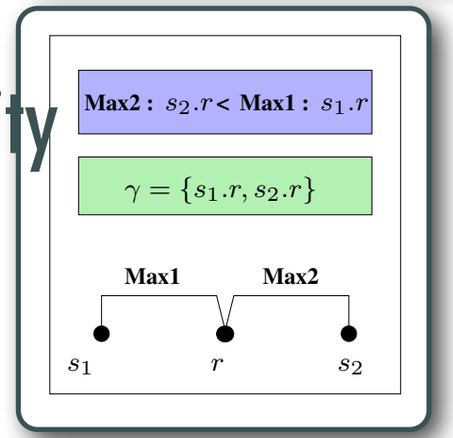

```
port type IntPort (int x)
port type ePort ()

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



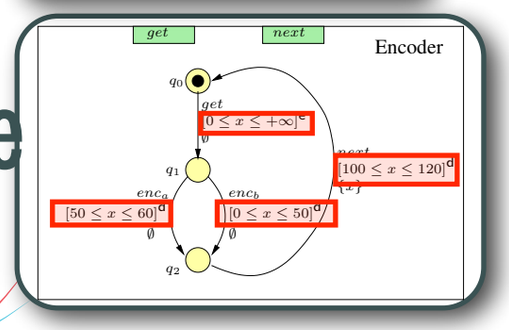

```
connector type Max (intPort p1, intPort p2)
   data int z
   define [ p1p2 ]

   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

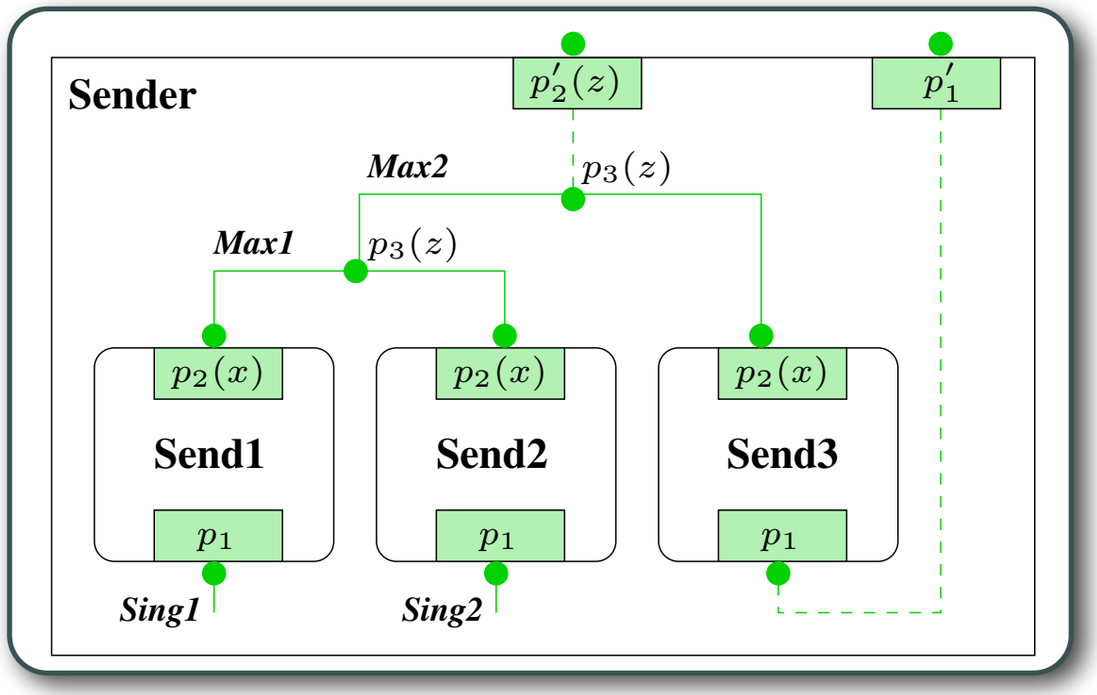



```
connector Max1 (s1, r)
connector Max2 (s2, r)

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

Time




```
atomic type Encoder
   export port intPort get
   export port intPort intPort next
   port intPort 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
```




```
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
```

GenoM to BIP

A template that produces the BIP model of any GenoM component specification

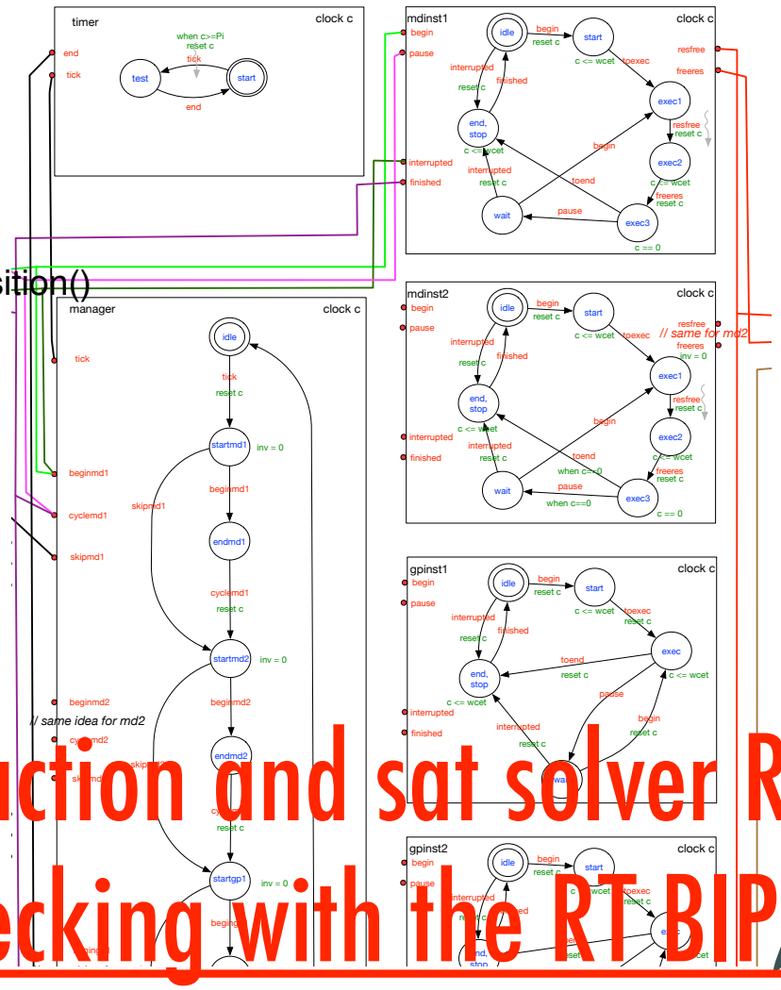
example:

/* plan timer */

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

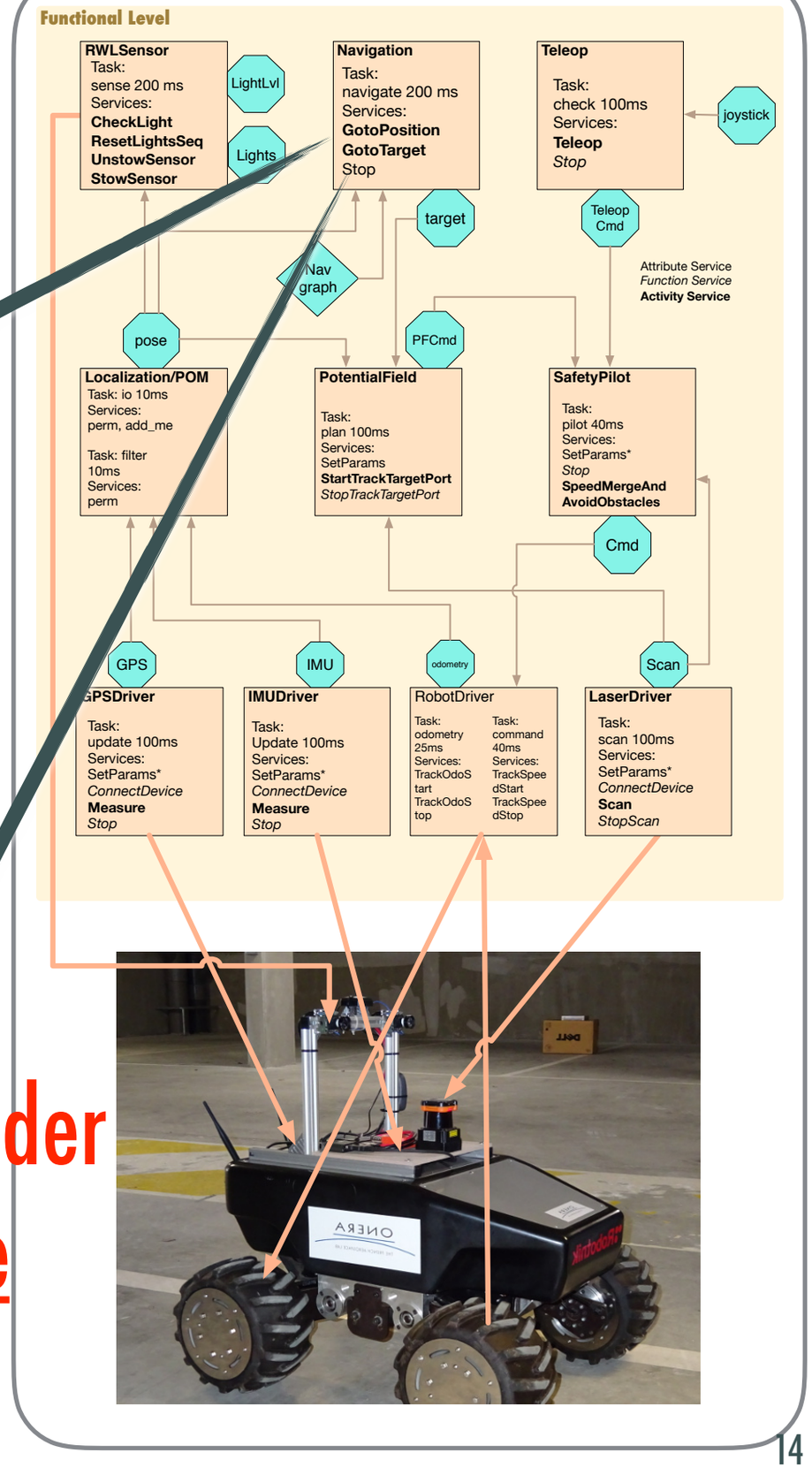
on tick
from loop to loop
provided (c>=200.0)
do { c = 0; }

end



Invariants extraction and sat solver RT D-Finder

Runtime Checking with the RT BIP Engine



Run Time Verification with RT BIP Engine

WCET violation in PotentialField:

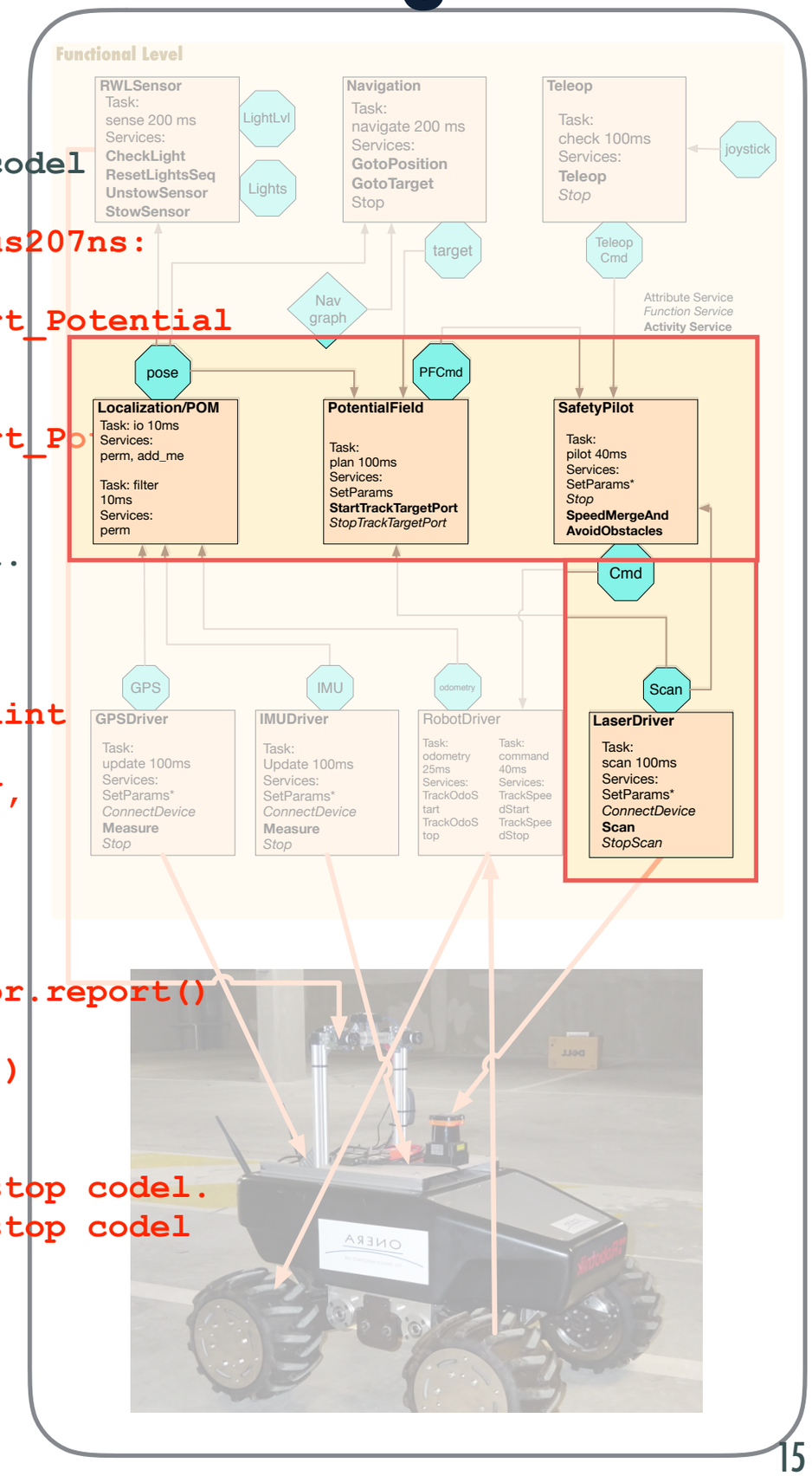
```
PF command: l 0.900901 m/s, a -22.167183 deg/s.
     [GenoM3] PotentialField Exiting
PotentialField_codel_service_StartTrackTargetPort_compute_speed codel
with ::PotentialField::write_cmd.
[BIP ENGINE]: WARNING: state #465945 and global time 58s764ms506us207ns:
violation of the following timing constraint
ROOT.Compound_PotentialField_.PotentialField_.StartTrackTargetPort_Potential
Field_inst_1:
[BIP ENGINE]:
ROOT.Compound_PotentialField_.PotentialField_.StartTrackTargetPort_Po
Field_inst_1 resume [ -INFTY, 58s763ms841us48ns ]
     [GenoM3] PotentialField Calling
PotentialField_codel_service_StartTrackTargetPort_write_cmd codel.
```

Cycle period violation in POM:

```
[BIP ENGINE]: WARNING: state #1905764 and global time
2min57s370ms722us604ns: violation of the following timing constraint
ROOT.pom.timer_filter:
[BIP ENGINE]:   ROOT.Compound_pom.timer_filter invariant [ -INFTY,
2min57s363ms458us605ns ]
```

Stop the robot when LaserDriver scan is delayed:

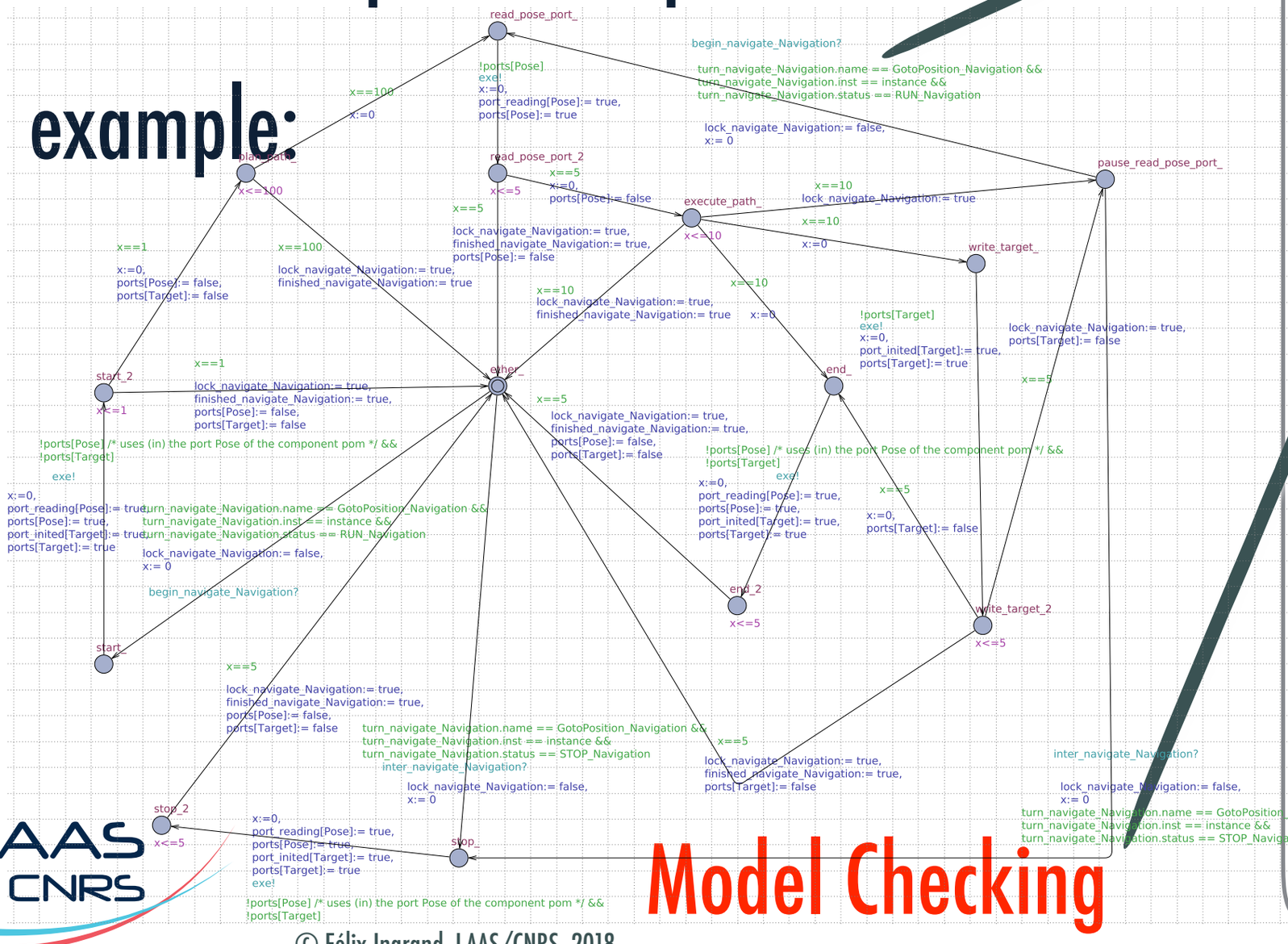
```
[BIP ENGINE]: state #165351: 1 interaction:
[BIP ENGINE]: [0] ROOT.Scan_Failed: SafetyPilot_Req_Stop() Monitor.report()
] 26s591ms324us108ns, +INFTY ]
[BIP ENGINE]: →choose [0] ROOT.Scan_Failed: SafetyPilot_Req_Stop()
Monitor.report() at global time 26s591ms324us109ns
...
[GenoM3] SafetyPilot Calling SafetyPilot_activity_MergeAndAvoid_stop codel.
[GenoM3] SafetyPilot Exiting SafetyPilot_activity_MergeAndAvoid_stop codel
with ::SafetyPilot::ether.
```

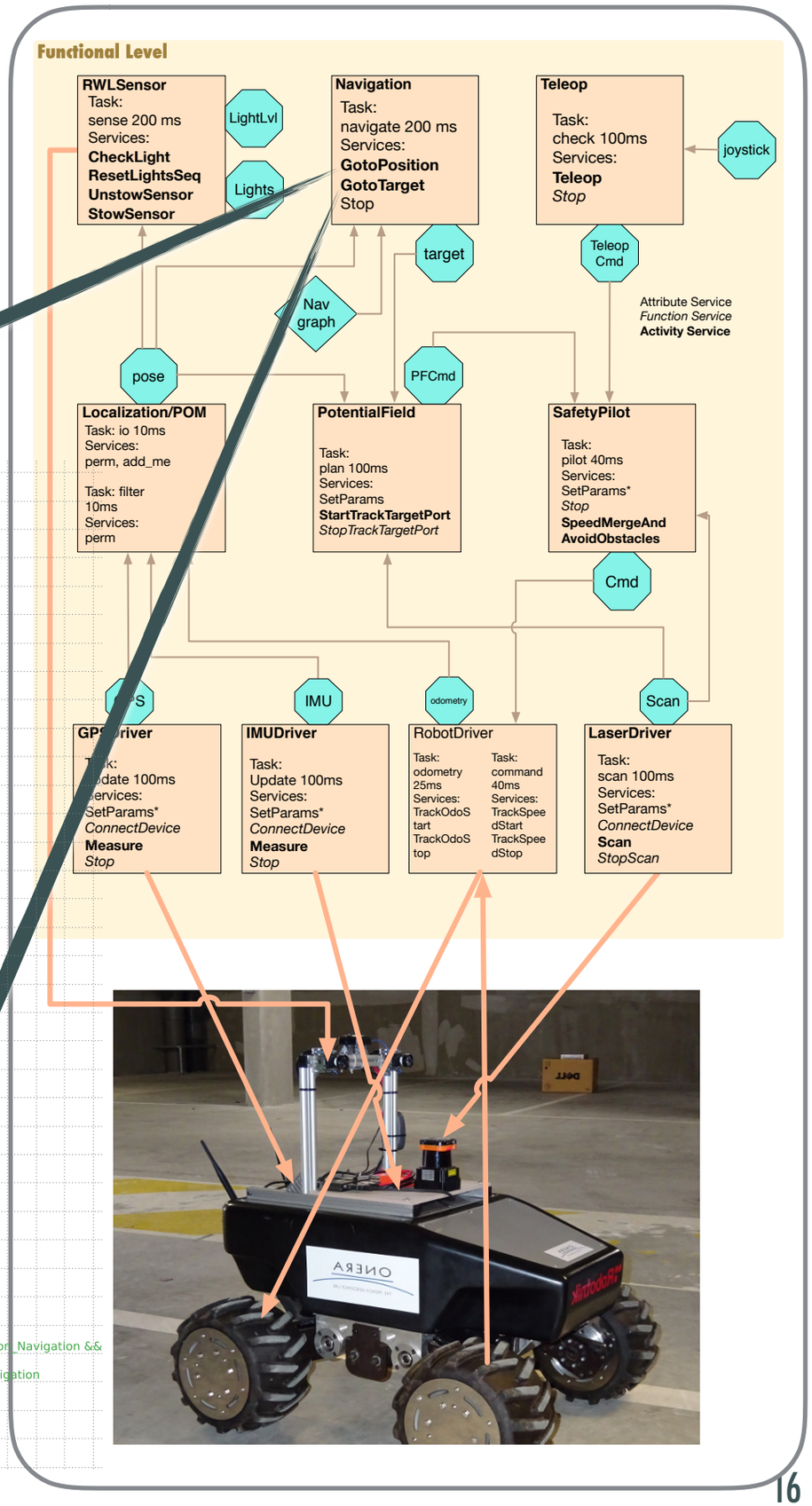


GenoM to UPPAAL

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

example:

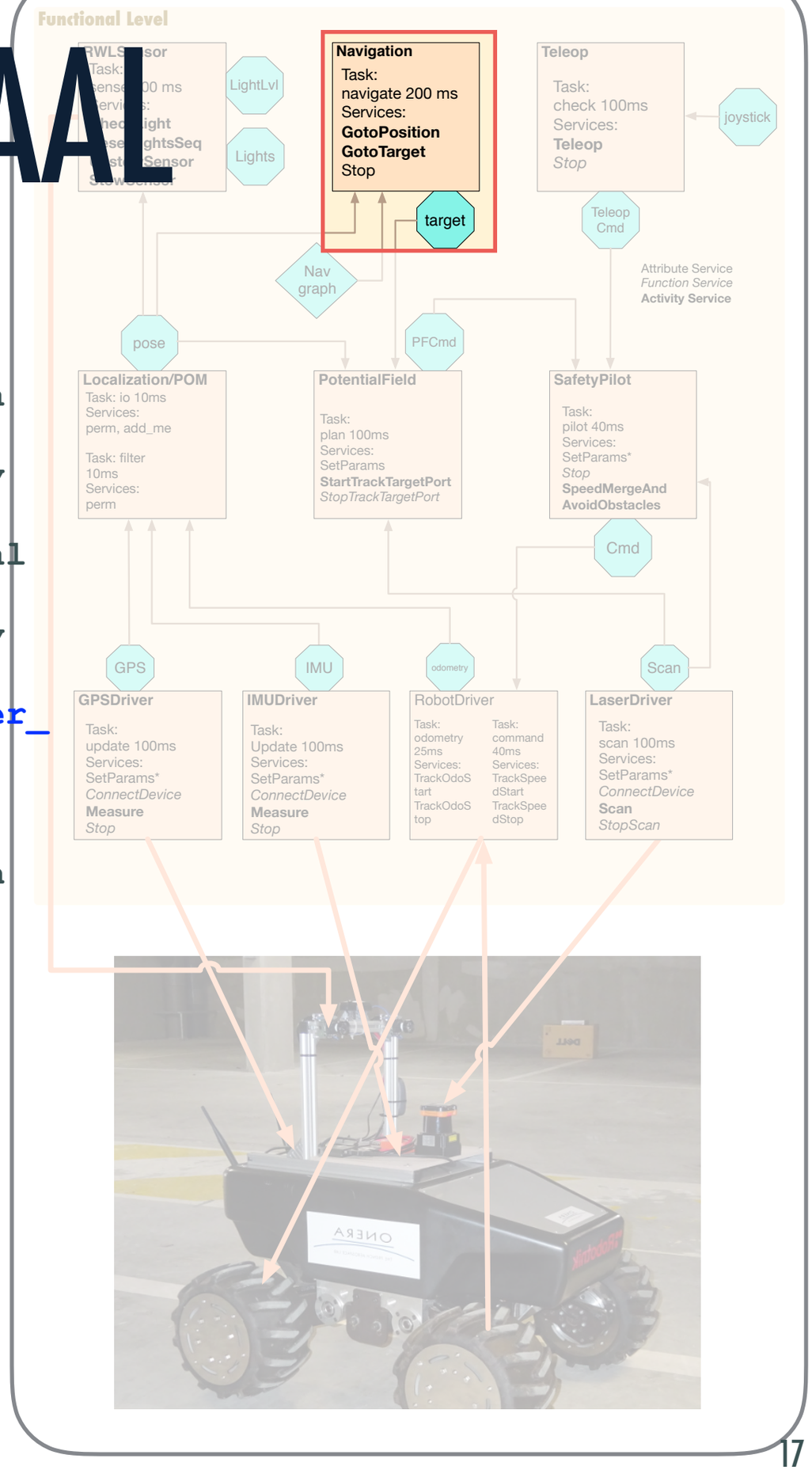




Model Checking

Verification with UPPAAL

```
/* All ports are properly written before being read */
A[] (ports_read[p] imply ports_write[p])


/* Navigation: bound between stop request and writing a
new target (to current pose) */
/* take advantage of the CT_Navigation clock reset only
once (when sending the stop request)
to verify a bounded-response property without additional
processes*/
/* bound = 202.5 ms, verification time 442.256s, memory
consumption ~1gb */
CT_Navigation.Stop_ --> (GotoPosition_1_Navigation.ether_
and CT_Navigation.x<=2025)

/* absence of (service) deadlock */
/* verification time 40 to 60s each, memory consumption
~250mb */
Man_navigate_Navigation.manage -->
Man_navigate_Navigation.start
Man_io_pom.manage --> Man_io_pom.start
Man_filter_pom.manage --> Man_filter_pom.start
Man_push_Localization.manage -->
Man_push_Localization.start
Man_plan_PotentialField.manage -->
Man_plan_PotentialField.start
Man_pilot_SafetyPilot.manage -->
Man_pilot_SafetyPilot.start
```



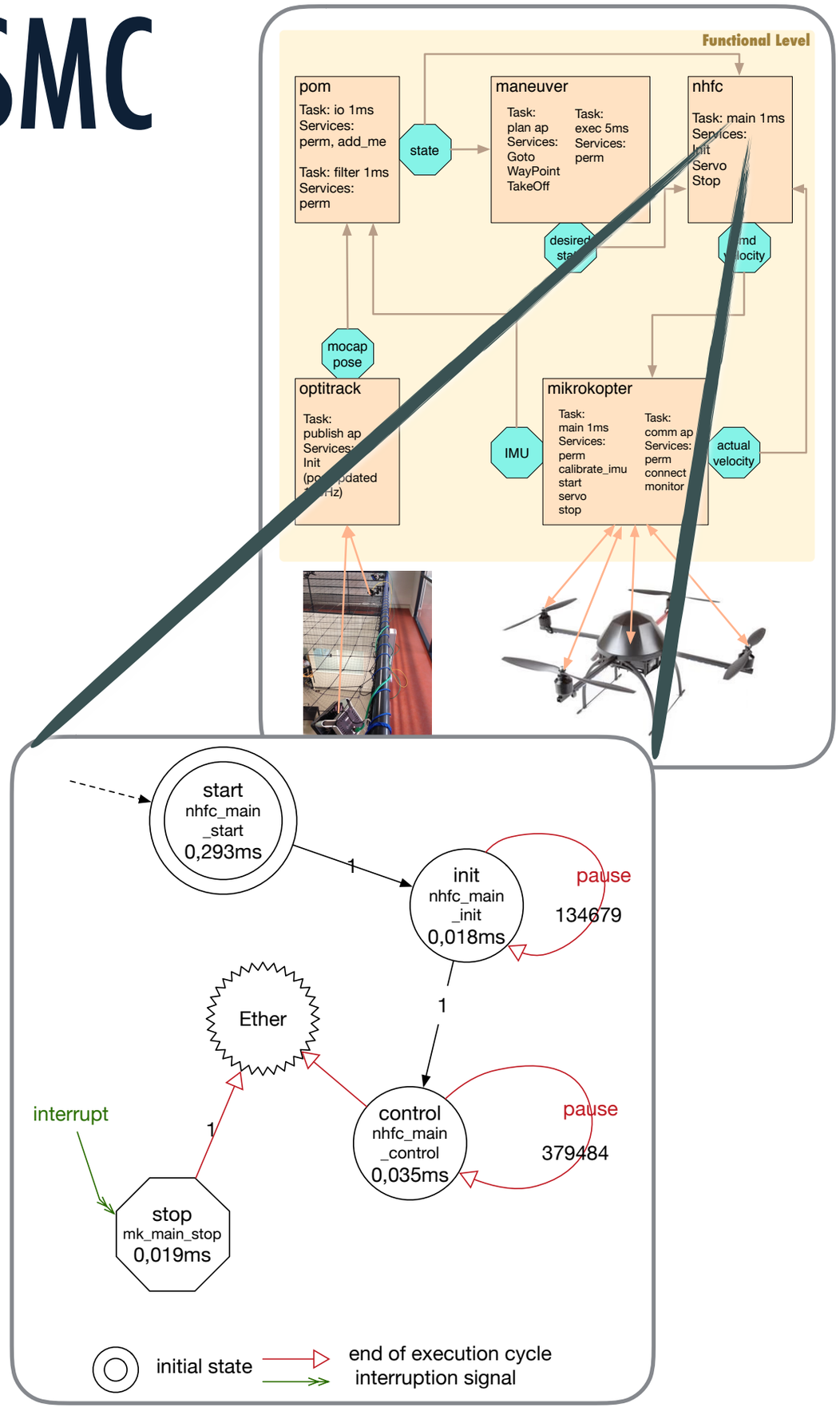
Verification with UPPAAL-SMC

- ✓ Statistical Model Checking 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.
```

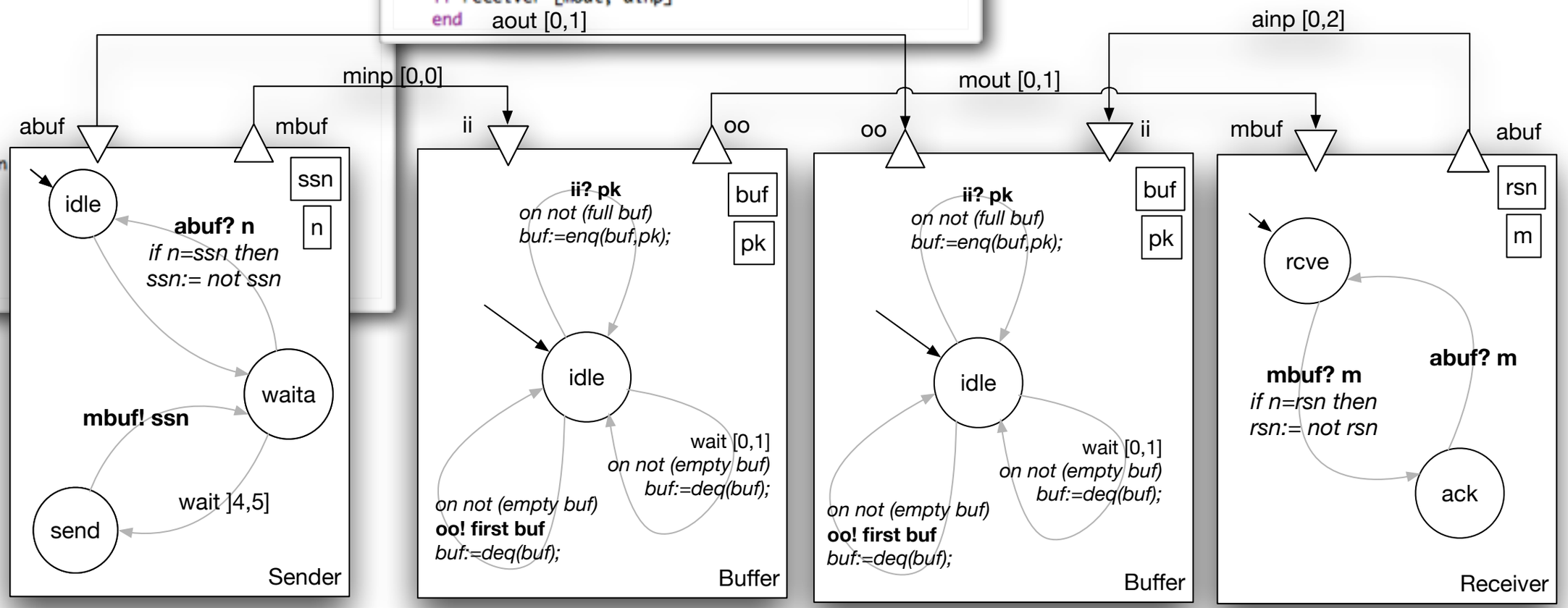
```
nhfc: nhfc_main_start called: 1 times, wcet: 0.000293.
nhfc: nhfc_main_init called: 134680 times, wcet: 0.000018.
nhfc: nhfc_main_control called: 379484 times, wcet: 0.000035.
nhfc: mk_main_stop called: 1 times, wcet: 0.000019.
```



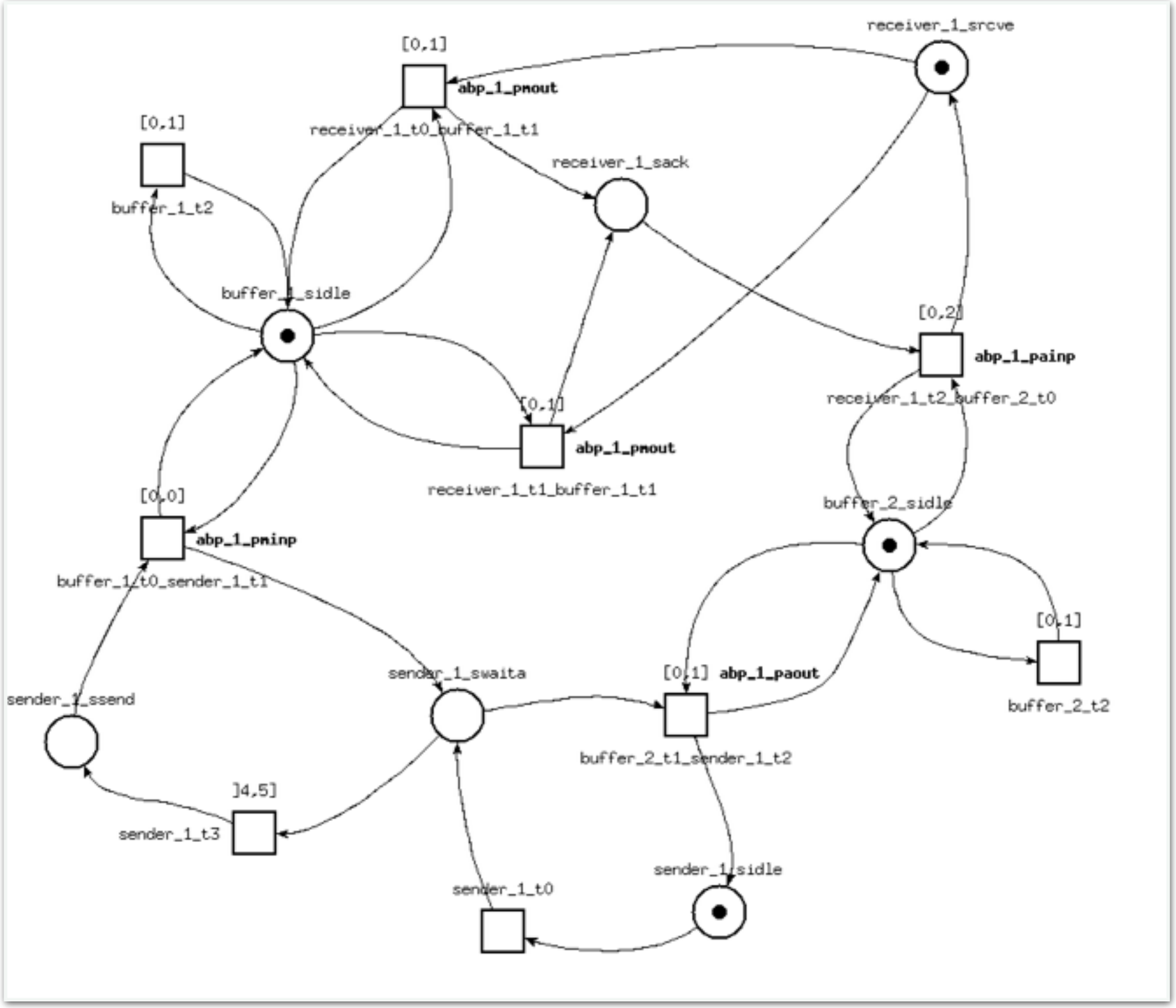
Fiacre Model example: Alternate Bit Protocol

```
/* 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
```

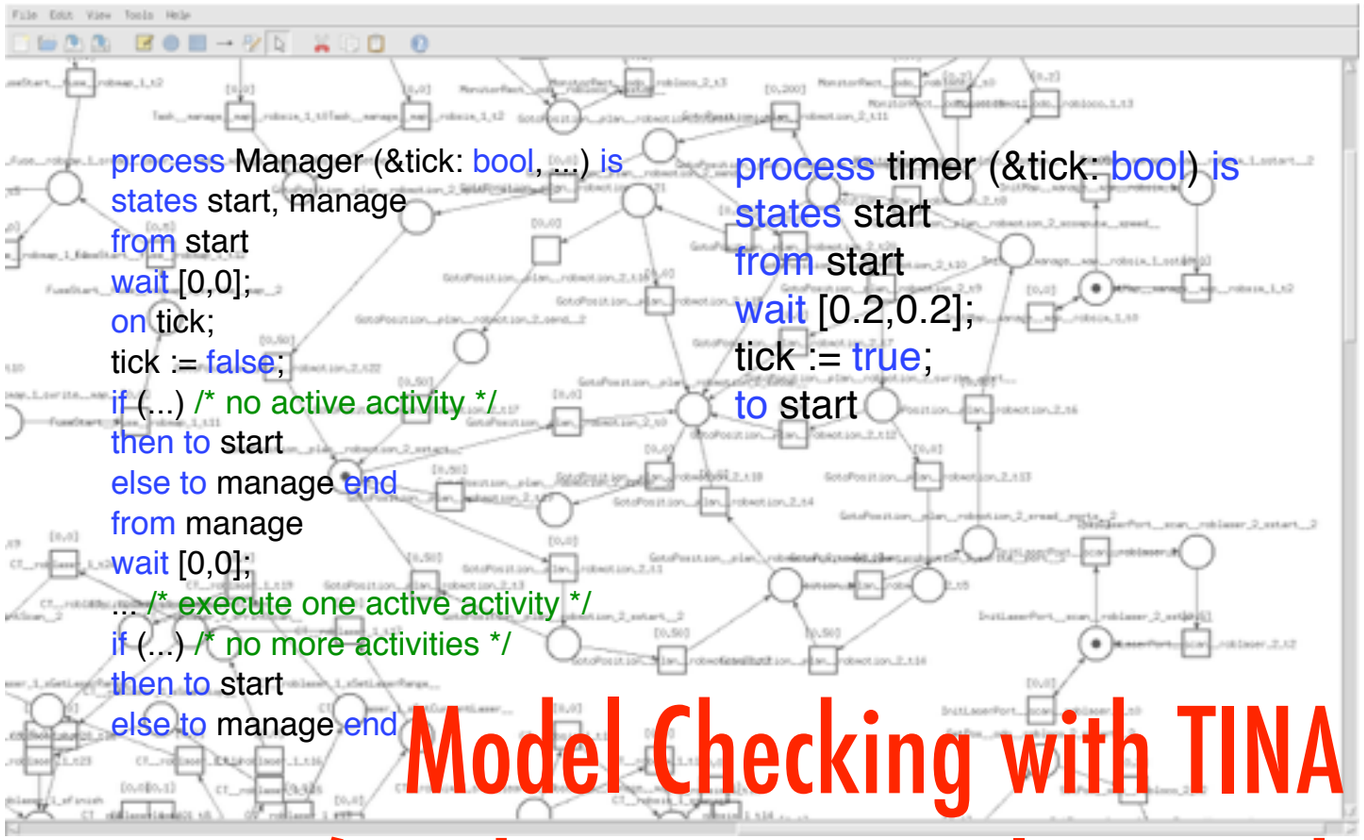


ABP FIACRE example automatically translated to Time Petri Net (TINA)



GenoM to Fiacre

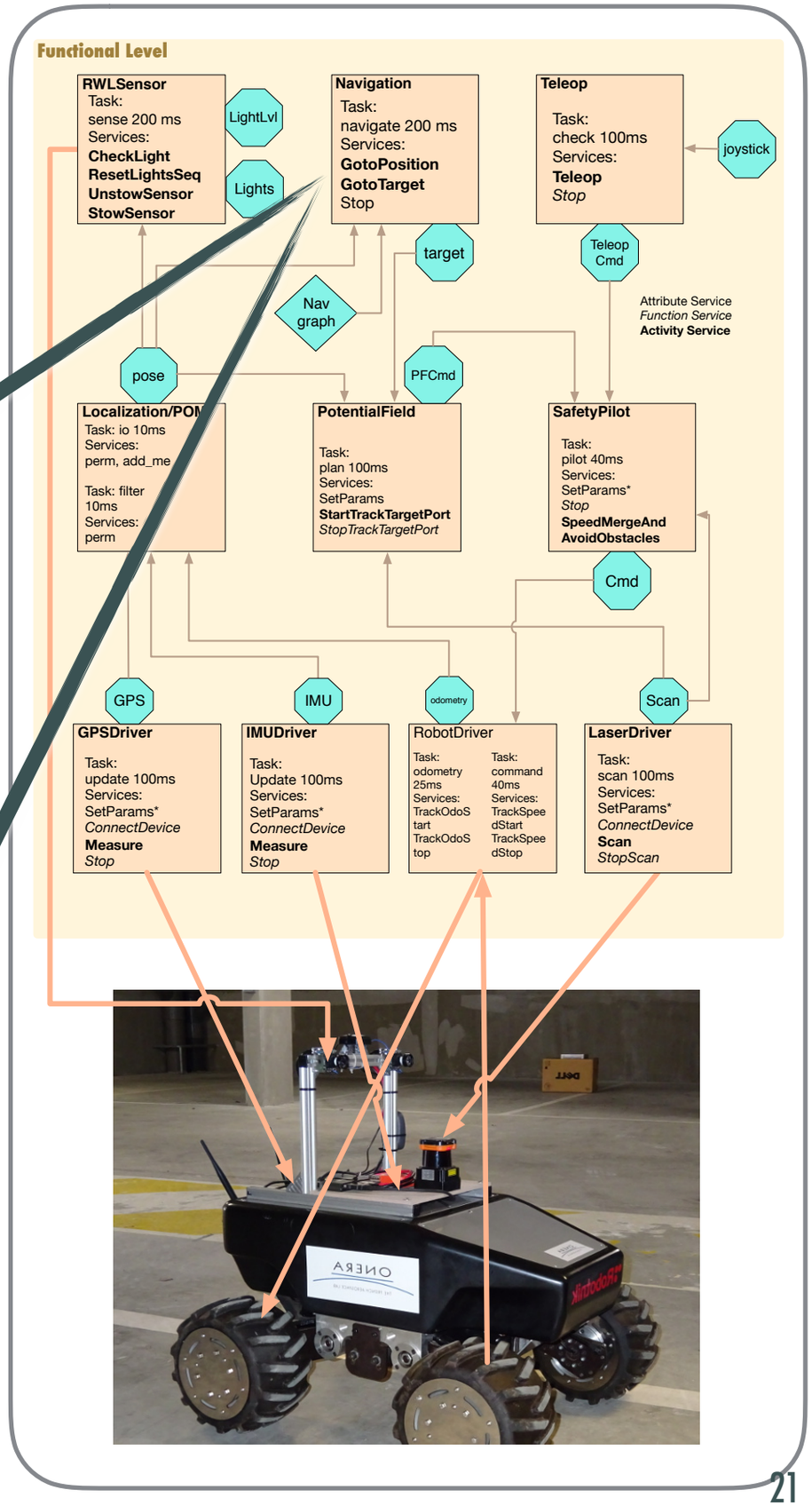
A template that produces the Fiacre model of any GenoM specification example:



```
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
```

```
process timer (&tick: bool) is
states start
from start
wait [0.2,0.2];
tick := true;
to start
```

Model Checking with TINA (on the TPN equivalent model)



Current GenoM V&V templates

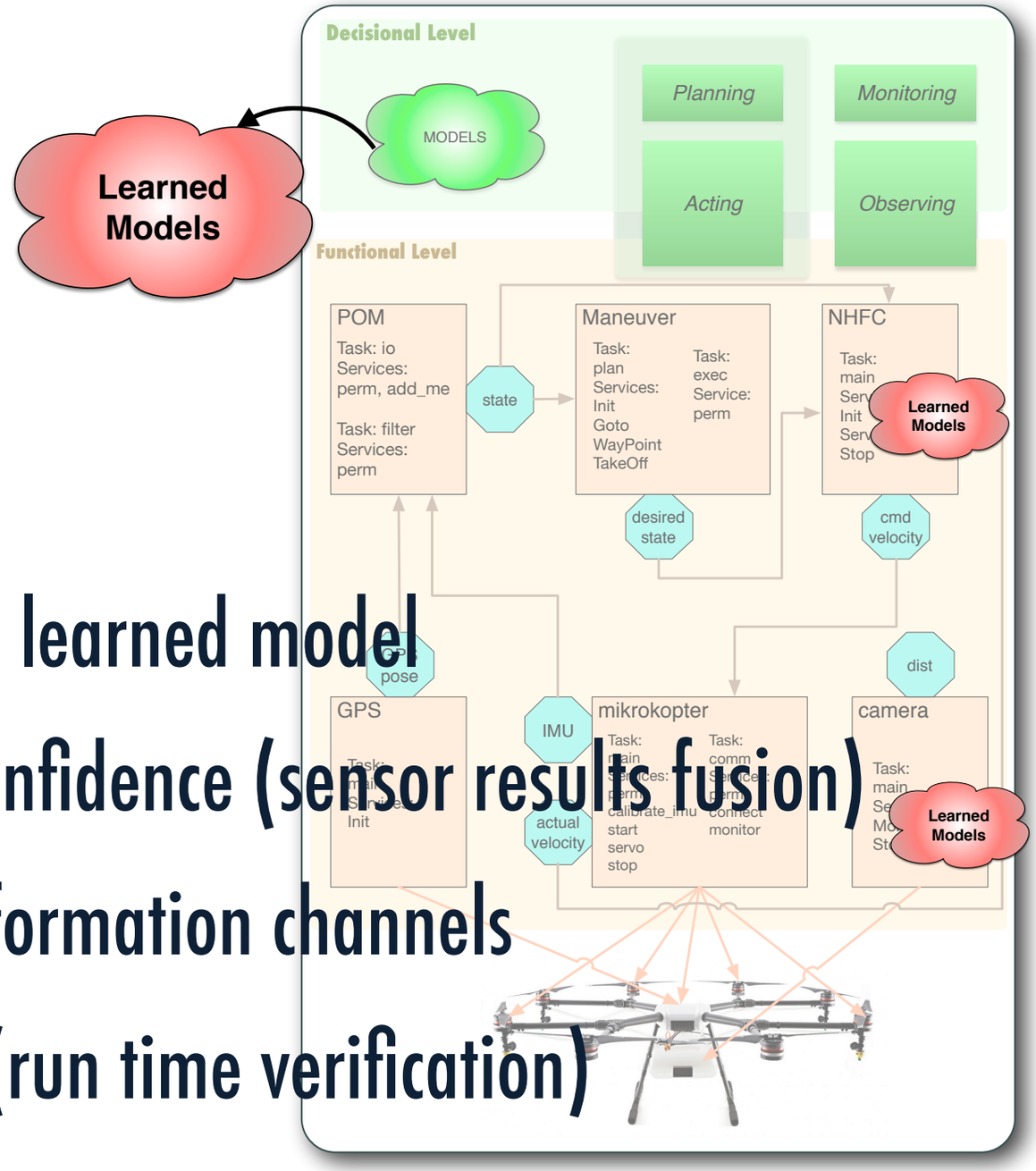
Formal Frameworks	Middleware: Offline	Online PocoLibs	Online ROS
BIP	- RT D-Finder	++	++
FIACRE	++	Under Dev	Under Dev
UPPAAL	+++		
UPPAAL SMC	++		

- The BIP model is complete, but has been a disappointment with respect to RT D-Finder
- The BIP-PocoLibs/ROS model for the BIP Engine is complete and functional
- The Fiacre template is complete and tested on numerous modules (model over multiple modules and ports communication), UPPAAL has a slight performance advantage.
- Between Fiacre and UPPAAL there are pros and cons (see M. Foughali's PhD)

LAAS CNRS

V&V of learned models…

- Machine learning is the new AI…
- Hard to extract a formal model… but we should try
- Proper environment modeling
- Properly characterize the bound of the learned model
- Use multiple sources to improve the confidence (sensor results fusion)
- Consistency checking over different information channels
- Safety bag around these components (run time verification)



[1] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," arXiv.org, 1606.06565v2, vol. cs.AI. 21-Jun-2016. http://arxiv.org/abs/1606.06565v2

[2] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Towards Verified Artificial Intelligence," arXiv.org, 1606.08514v3 vol. cs.AI. 28-Jun-2016. http://arxiv.org/abs/1606.08514v3

LAAS CNRS

Conclusion

- When there are models... there is hope!
- Adapt the model and the V&V techniques
- Try to keep the overall consistency
- AI components are mostly OK (wrt formal V&V)
- V&V and certification of “learned model” based components remain a challenge

LAAS CNRS

Research agenda

- FYI: NHTSA just allowed testing with cars without steering wheels... (level 4)
- Deeper model (codel arguments, SDI, algo, check the codel, etc) & Run Time Verification
- Better linked models between functional level and decisional level (Planning/Acting/Monitoring)
- Address V&V of learned models
- Human in the loop (uncontrollable model)

LAAS CNRS

Thanks to

Verimag: Jacques Combaz, Saddek Bensalem

LAAS: Anthony Mallet, Mohammed Foughali, Bernard Berthomieu,
Silvano Dal Zilio, Pierre Emanuel Hladik

LAAS CNRS

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