An Architecture for Robust and Fault Tolerant Autonomous Robots

Raja Chatila, Sara Fleury, Matthieu Gallien, Matthieu Herrb,
Felix Ingrand, Benjamin Lussier, David Powell, Fréderic Py

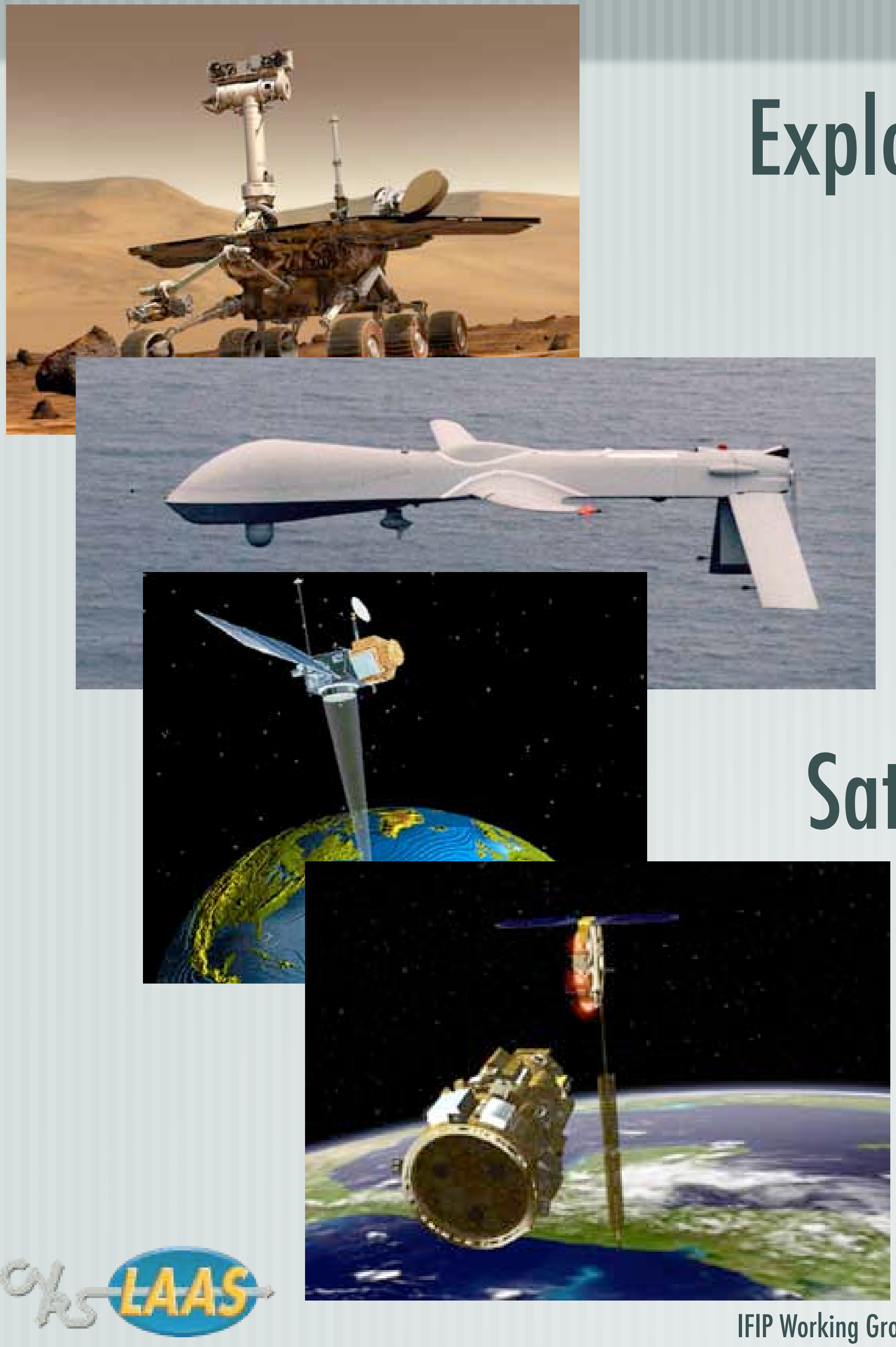
LAAS - CNRS
Toulouse, France

Autonomous Systems

Lama

Dala

Autonomous Systems



Exploration Rovers

Drones

Satellites

Space Probes

Lama

Dala

Interacting with humans

Autonomous Systems

Autonomous Systems

Companion Robots

Interacting with humans

Autonomous Systems

Service Robots

Companion Robots

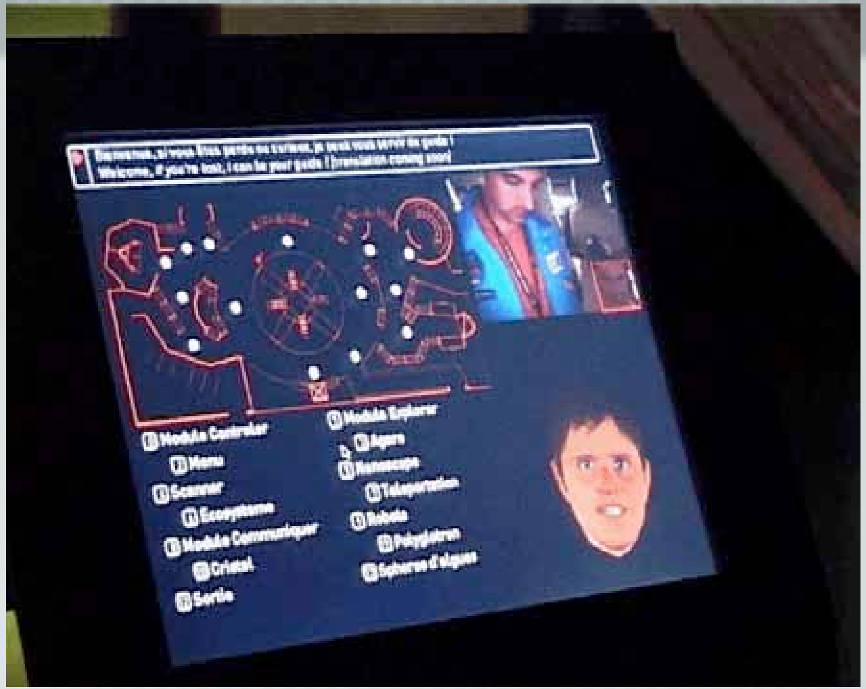
Interacting with humans

Autonomous Systems

Companion Robots

Service Robots

Tour Robots



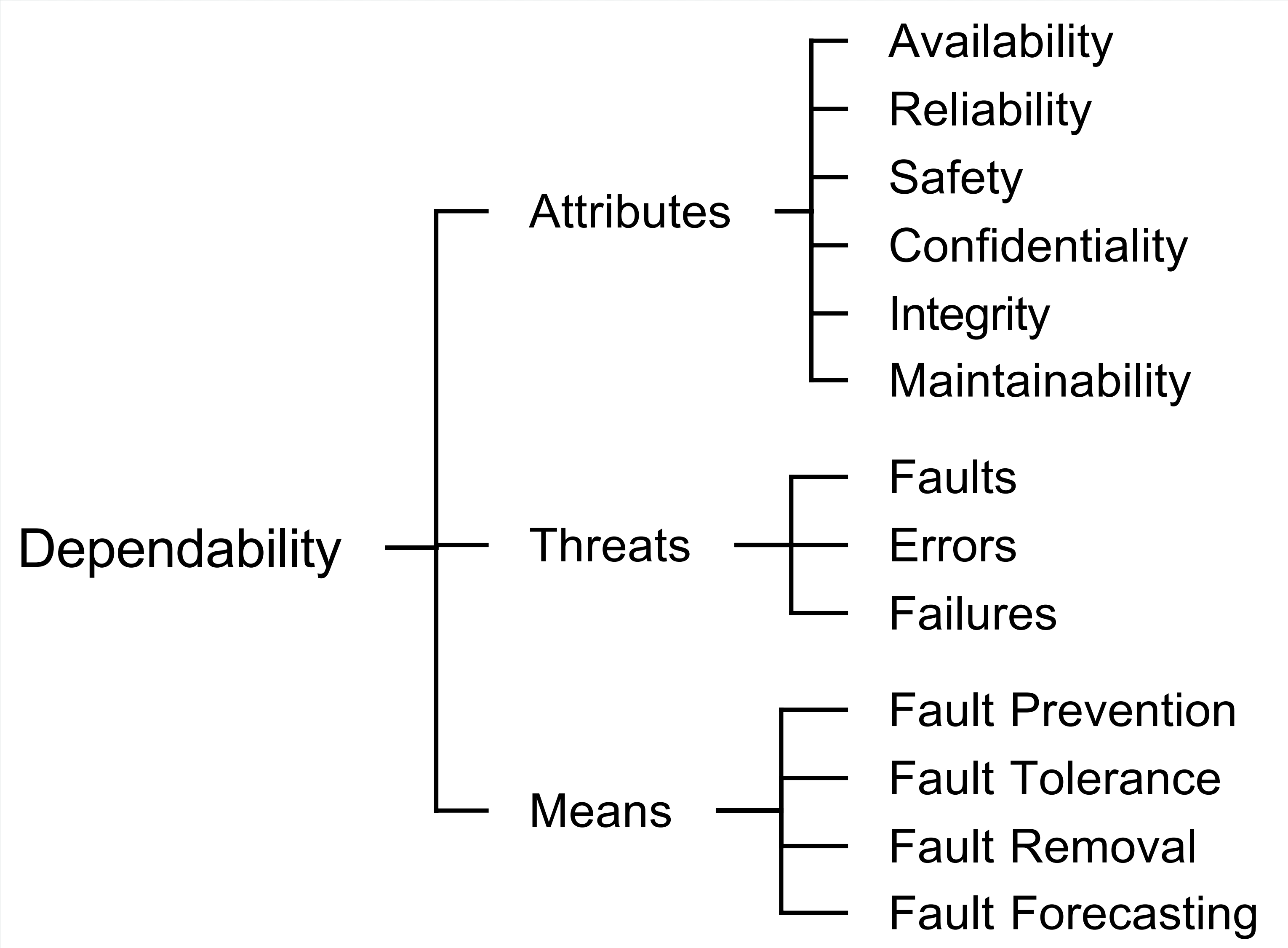
Bienvenue, si vous êtes perdu ou curieux, je peux vous servir de guide !
Welcome, if you're lost, I can be your guide ! [translation coming soon]

The problem

To improve the dependability of Autonomous robots and systems

- Large number of functional subsystems
  - Sensors/Effectors
- Decisional capabilities
  - planning/scheduling, supervision and plan execution control
- Evolve in the real world...

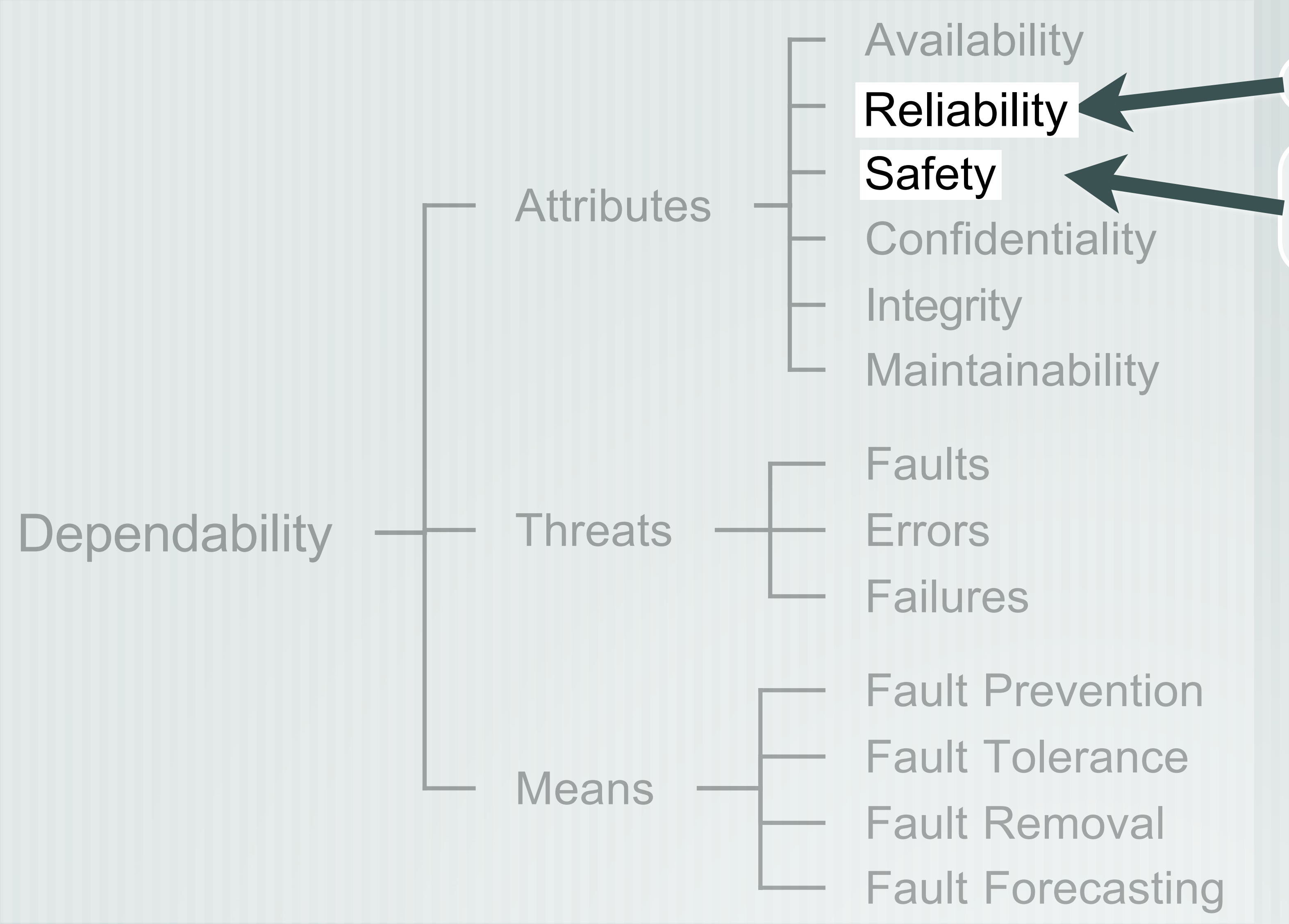
Dependability



- Dependability
  - Attributes
    - Availability
    - Reliability
    - Safety
    - Confidentiality
    - Integrity
    - Maintainability
  - Threats
    - Faults
    - Errors
    - Failures
  - Means
    - Fault Prevention
    - Fault Tolerance
    - Fault Removal
    - Fault Forecasting

[ALR 04] A. Avizienis, J.C. Laprie & B. Randell, Dependability and its Threats : A Taxonomy. 18th IFIP World Congress, 2004

Dependability

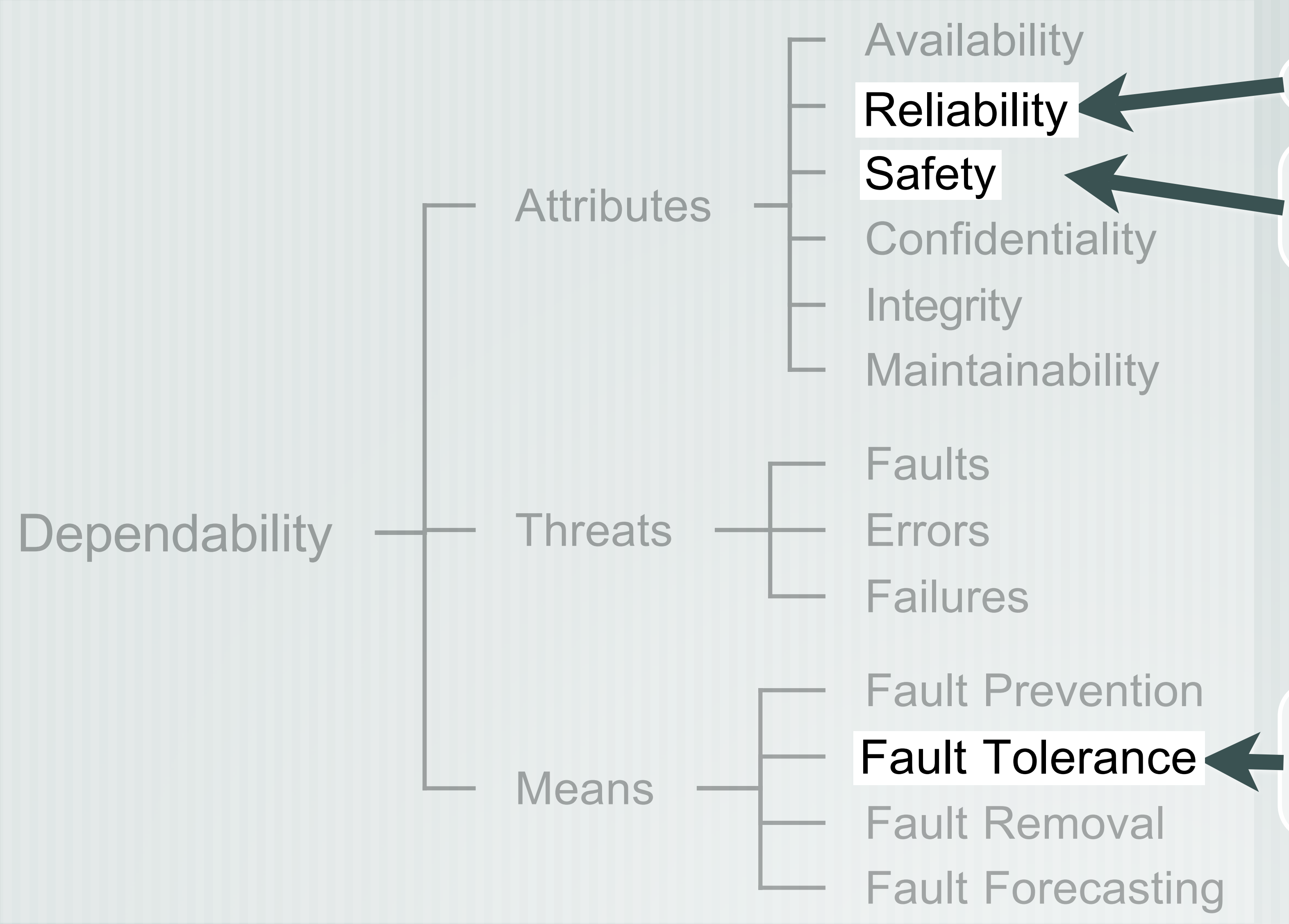


continuity of correct service

absence of catastrophic consequences on the user(s) and the environment

[ALR 04] A. Avizienis, J.C. Laprie & B. Randell, Dependability and its Threats : A Taxonomy. 18th IFIP World Congress, 2004

Dependability



Dependability

Attributes

Availability

Reliability

Safety

Confidentiality

Integrity

Maintainability

Threats

Faults

Errors

Failures

Means

Fault Prevention

Fault Tolerance

Fault Removal

Fault Forecasting

continuity of correct service

absence of catastrophic consequences on the user(s) and the environment

means to avoid service failures in the presence of faults

[ALR 04] A. Avizienis, J.C. Laprie & B. Randell, Dependability and its Threats : A Taxonomy. 18th IFIP World Congress, 2004

Objectives

- To offer some guarantees on the dependability of autonomous systems (reliability and safety)
- Choice of architecture : Hierarchical Architecture
- Mean : Online execution control (fault tolerance)

Why an architecture?

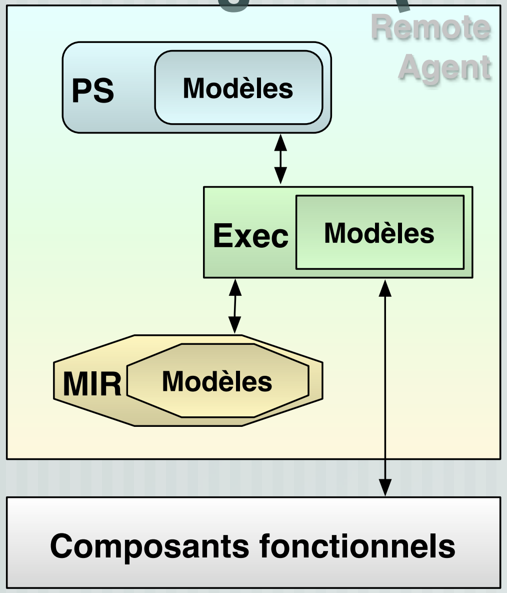
- Robots are complex systems
  - numerous sensors and effectors
- Various type of processing
  - functional / decisional
  - real time / exponential complexity
- Sharing information and codes
  - interoperability

Properties

- Programmability
  - multiple environments or tasks,
  - different abstract levels
- Adaptability
- Reactivity
- Consistent behavior
- Extensibility / Reusability
- Robustness / Dependability

Architectures

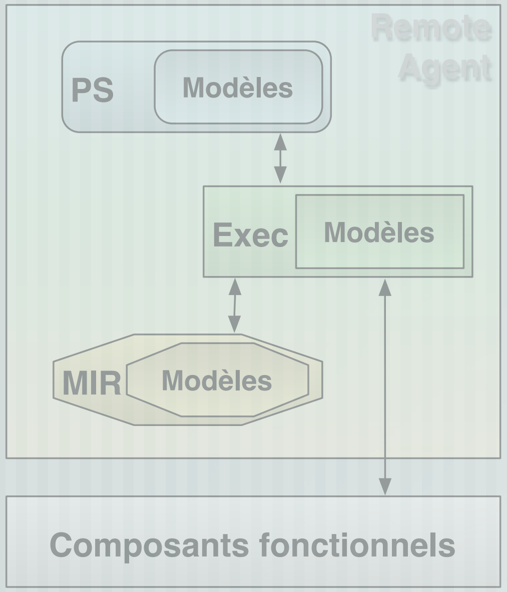
Remote Agent (Nasa)

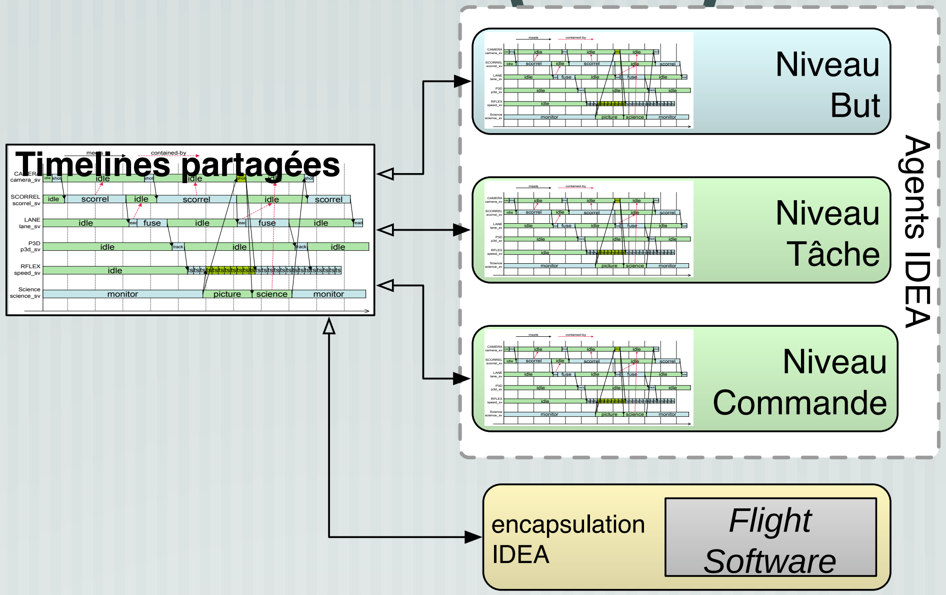


[Bernard 00] D. Bernard et al., Remote Agent Experiment.
Rapport technique Nasa ARC & JPL, 2000

Architectures

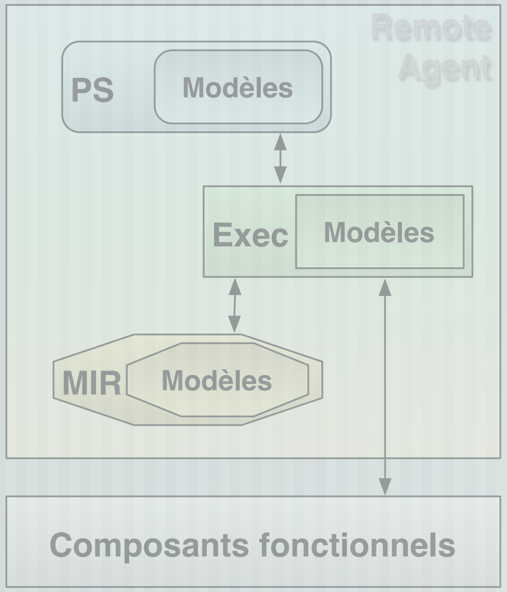
IDEA (Nasa)

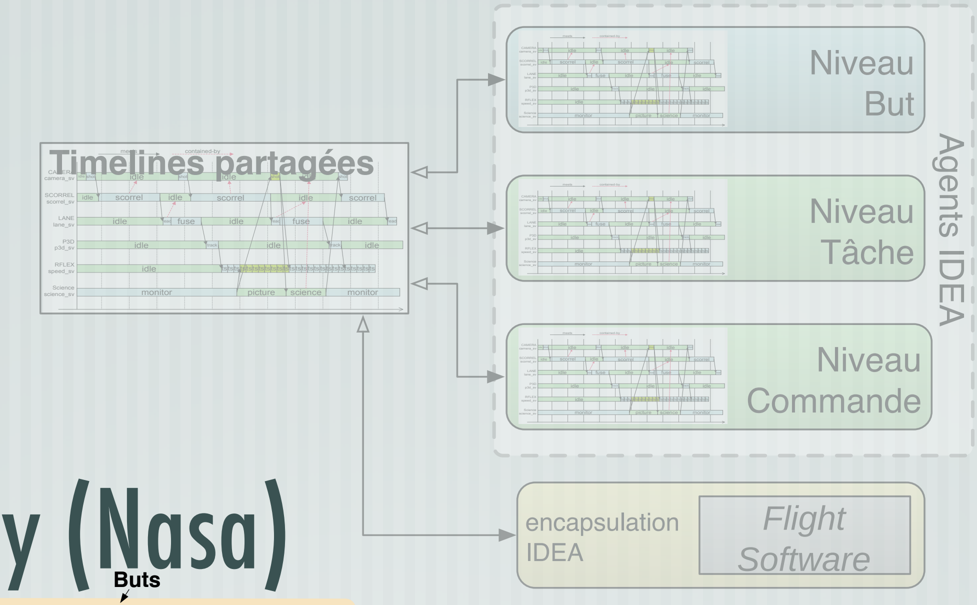




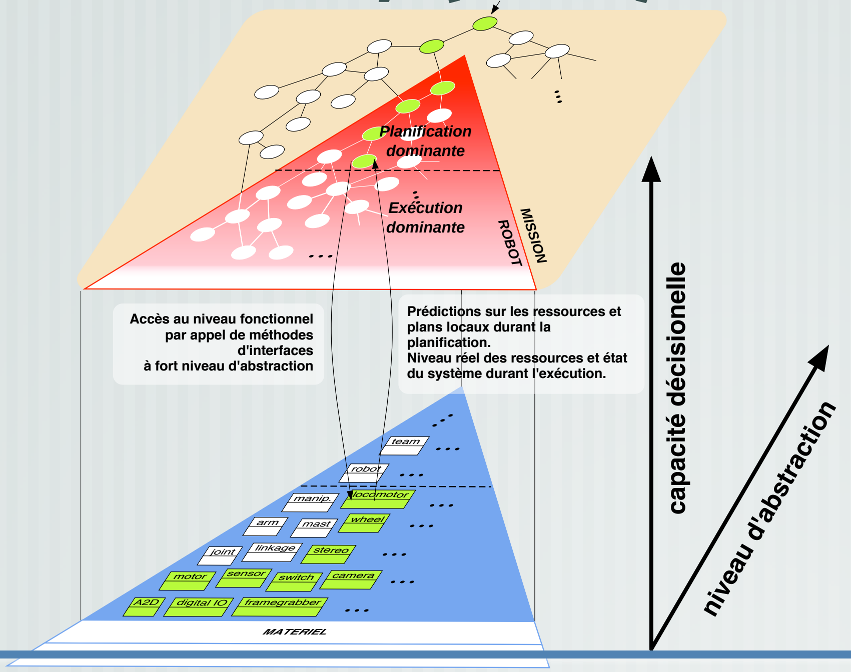
[Muscettola 02] N. Muscettola et al., IDEA : Planning at the Core of Autonomous Reactive Agents.
3rd Int. NASA Workshop on Planning & Scheduling for Space, 2002

Architectures



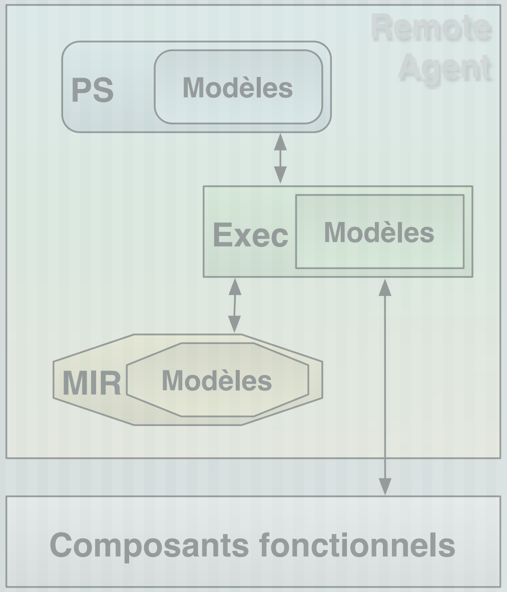


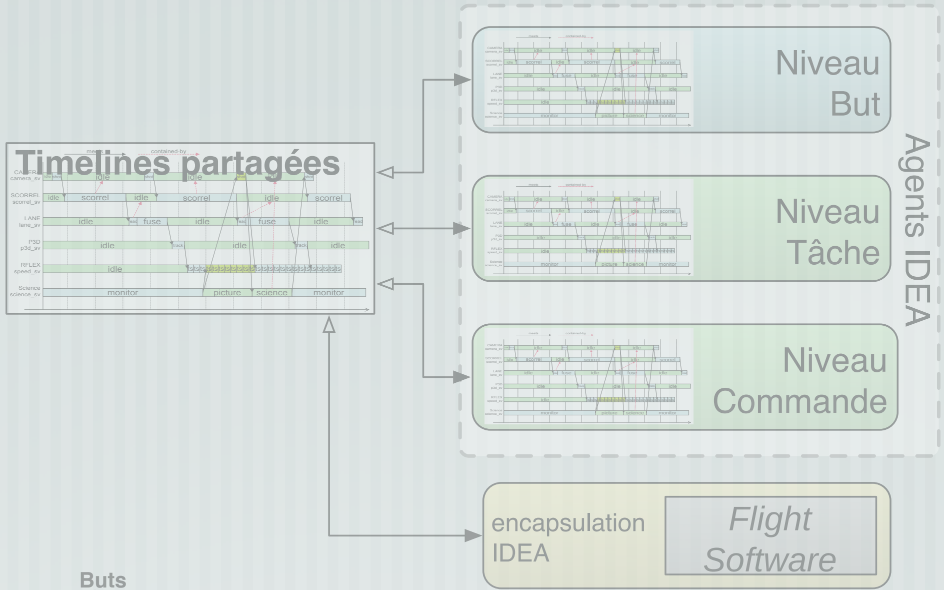
CLARAty (Nasa)

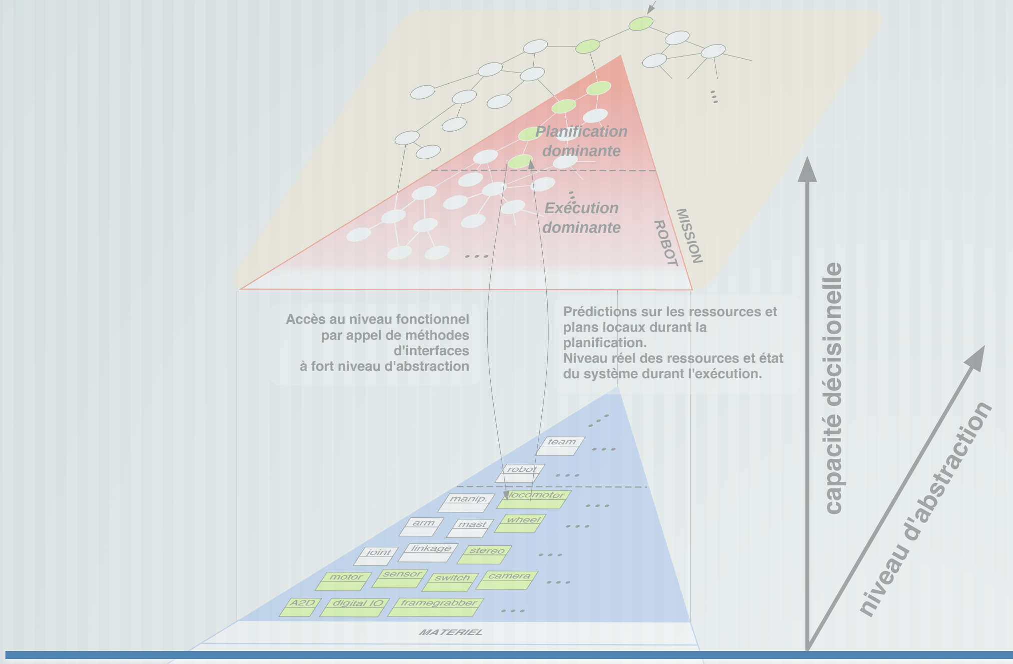

Buts
Planification dominante
Exécution dominante
MISSION
ROBOT
Accès au niveau fonctionnel par appel de méthodes d'interfaces à fort niveau d'abstraction
Prédictions sur les ressources et plans locaux durant la planification.
Niveau réel des ressources et état du système durant l'exécution.
team
robot
manip.
locomotor
arm
mast
wheel
joint
linkage
stereo
motor
sensor
switch
camera
A2D
digital IO
framegrabber
MATERIEL
capacité décisionelle
niveau d'abstraction

[NWBSE 03] I.A. Nesnas et al., CLARAty and Challenges of Developing Interoperable Robotic Software. IROS 2003

Architectures



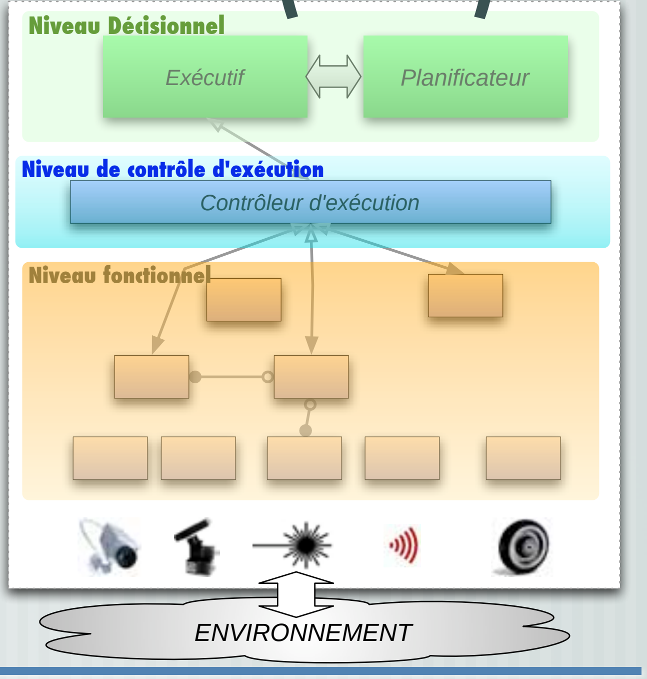




Accès au niveau fonctionnel par appel de méthodes d'interfaces à fort niveau d'abstraction

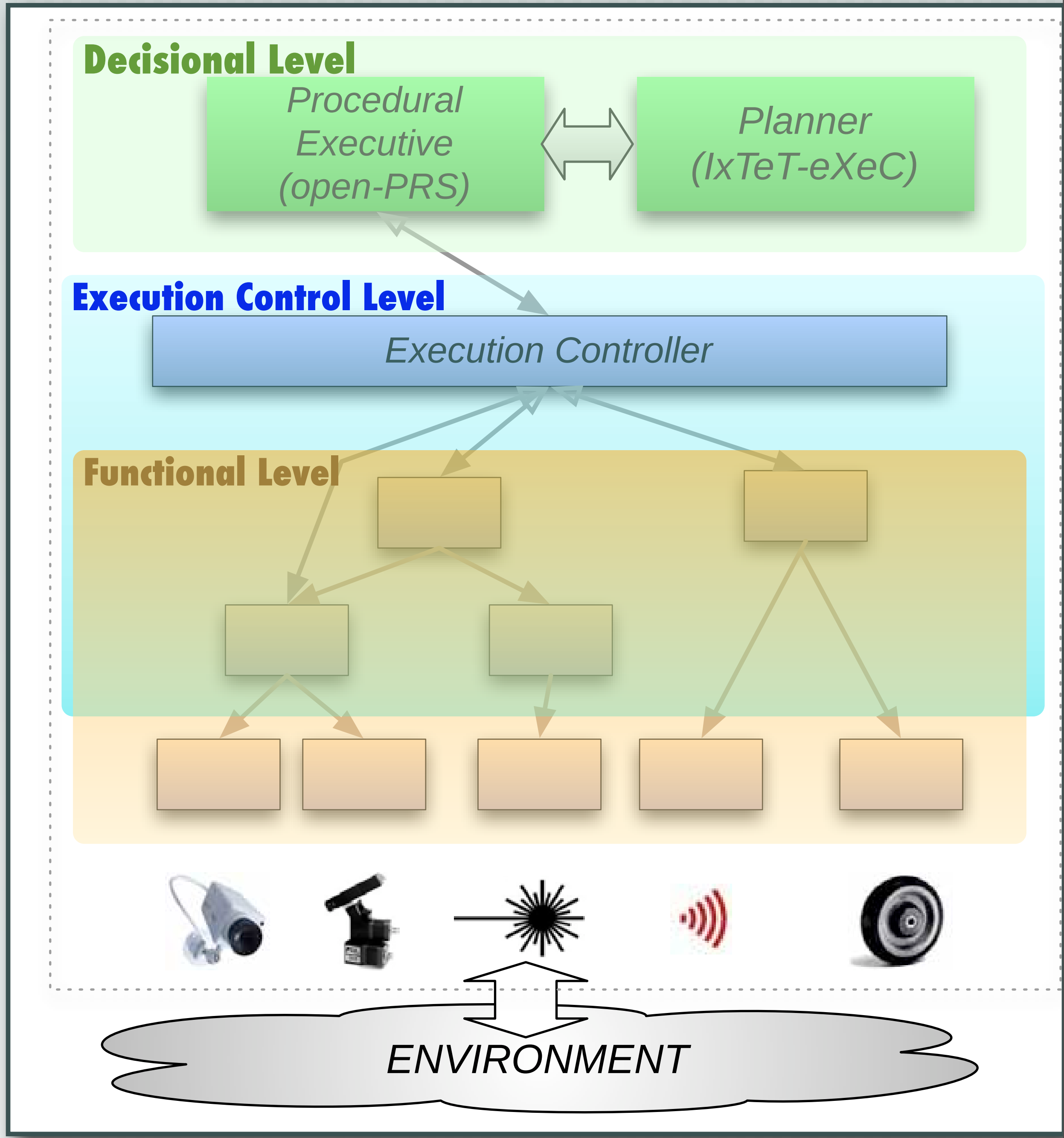
Prédictions sur les ressources et plans locaux durant la planification.
Niveau réel des ressources et état du système durant l'exécution.

LAAS (LAAS)



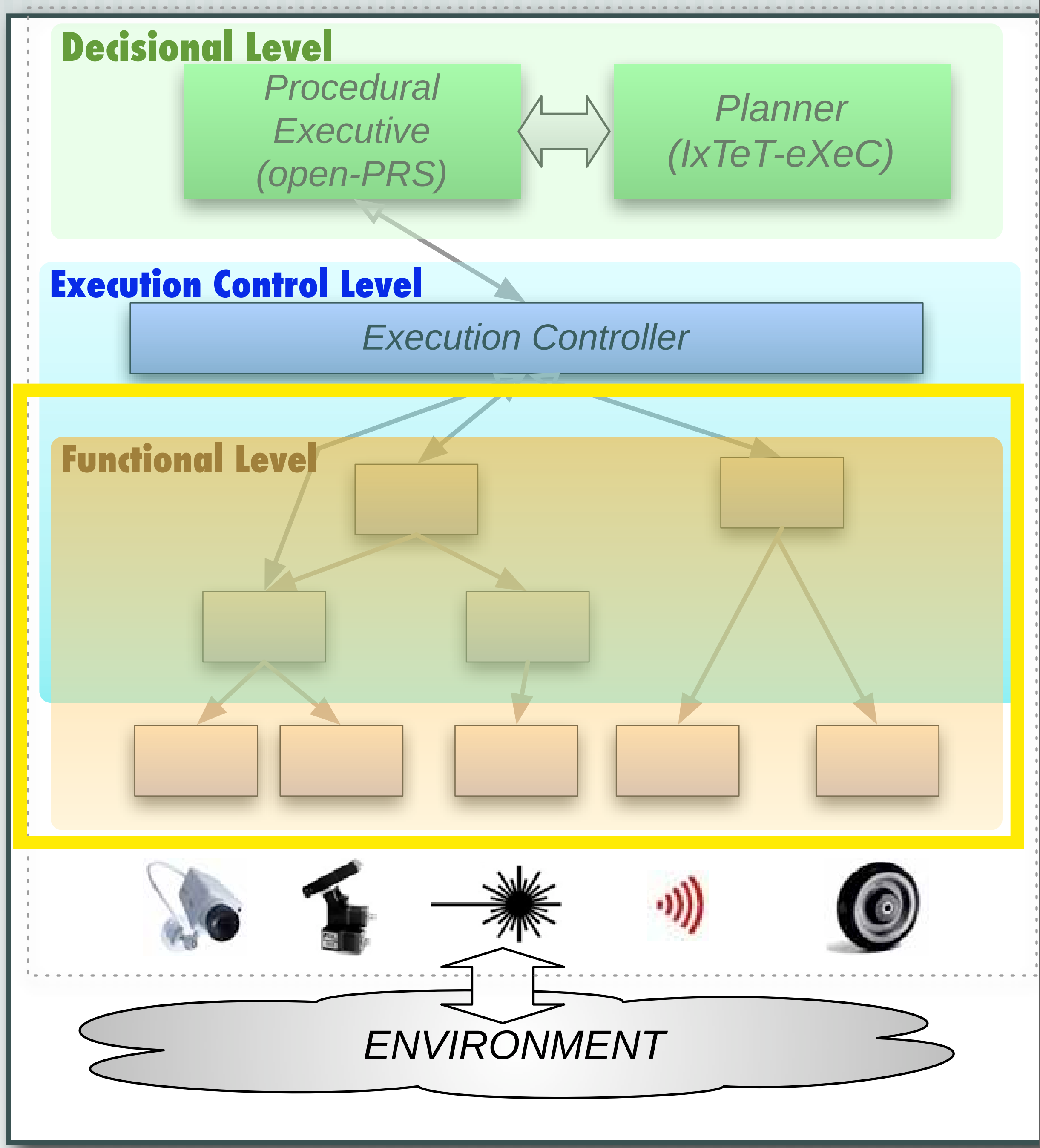
The LAAS Architecture

The LAAS Architecture

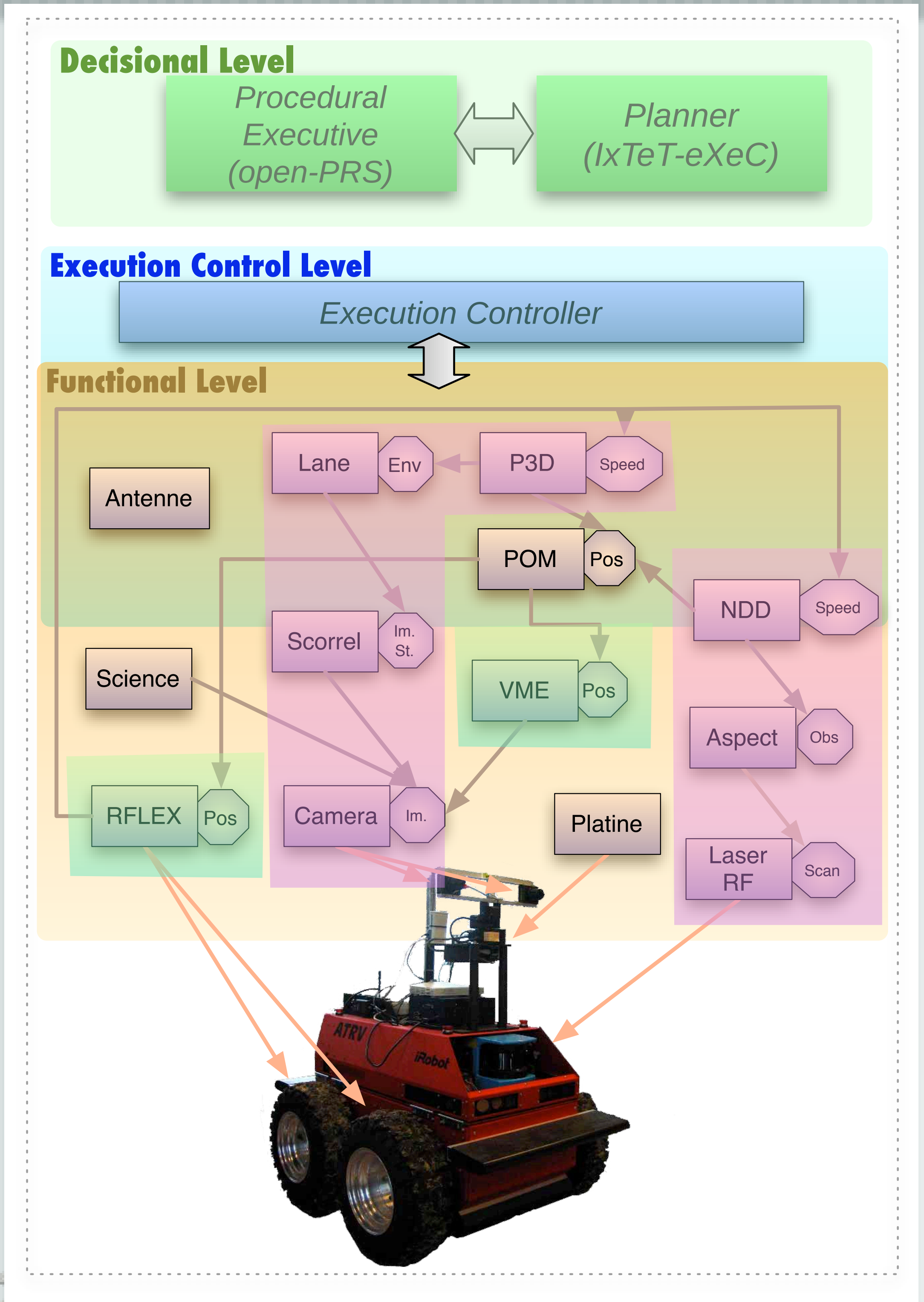


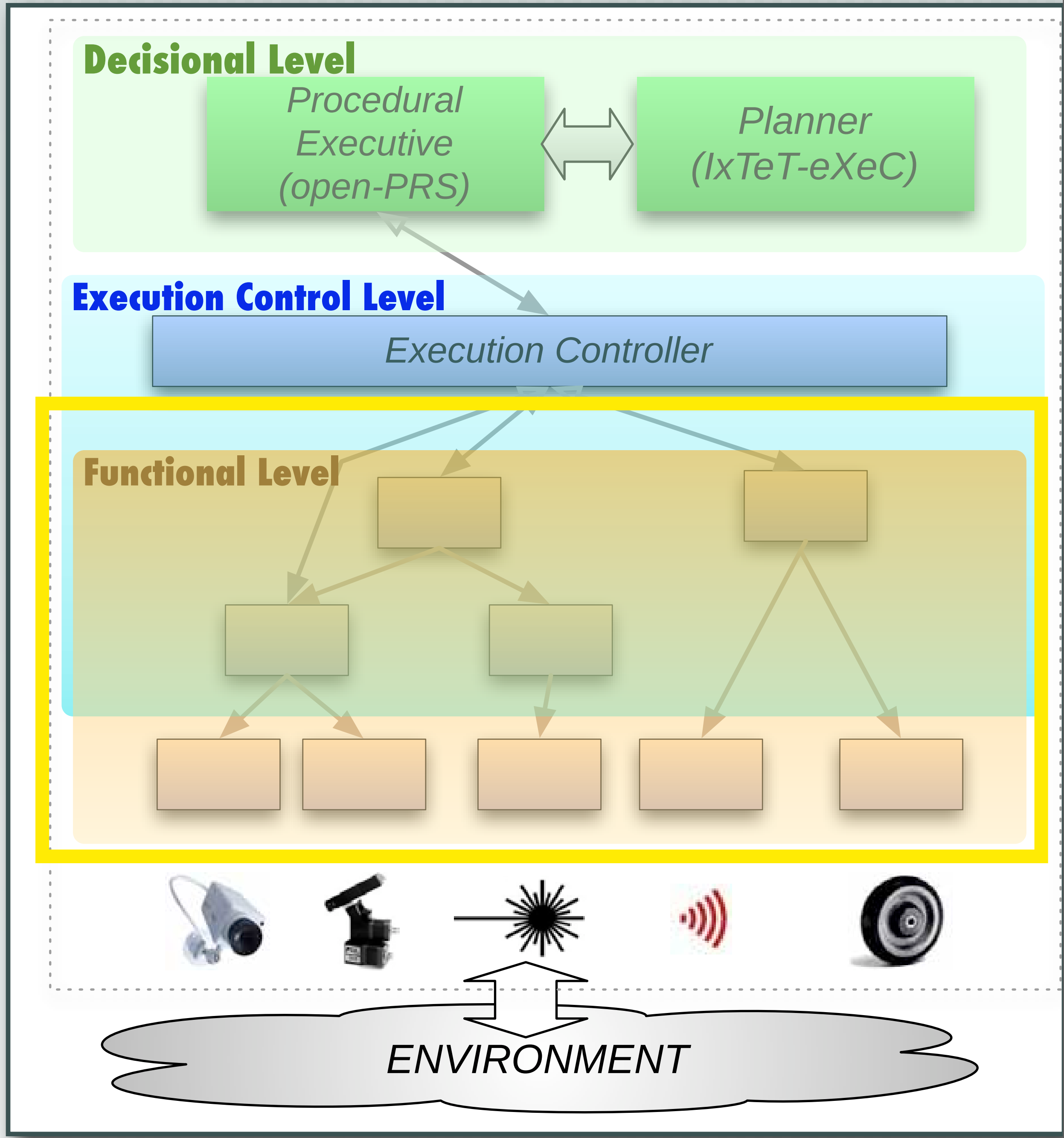
Functional Level

- GenoM independant modules corresponding to a group of functionalities.
- Each module provides a service
- Real Time aspect
- Algorithm are broken down in pieces
- Each task has its own priority/ frequency

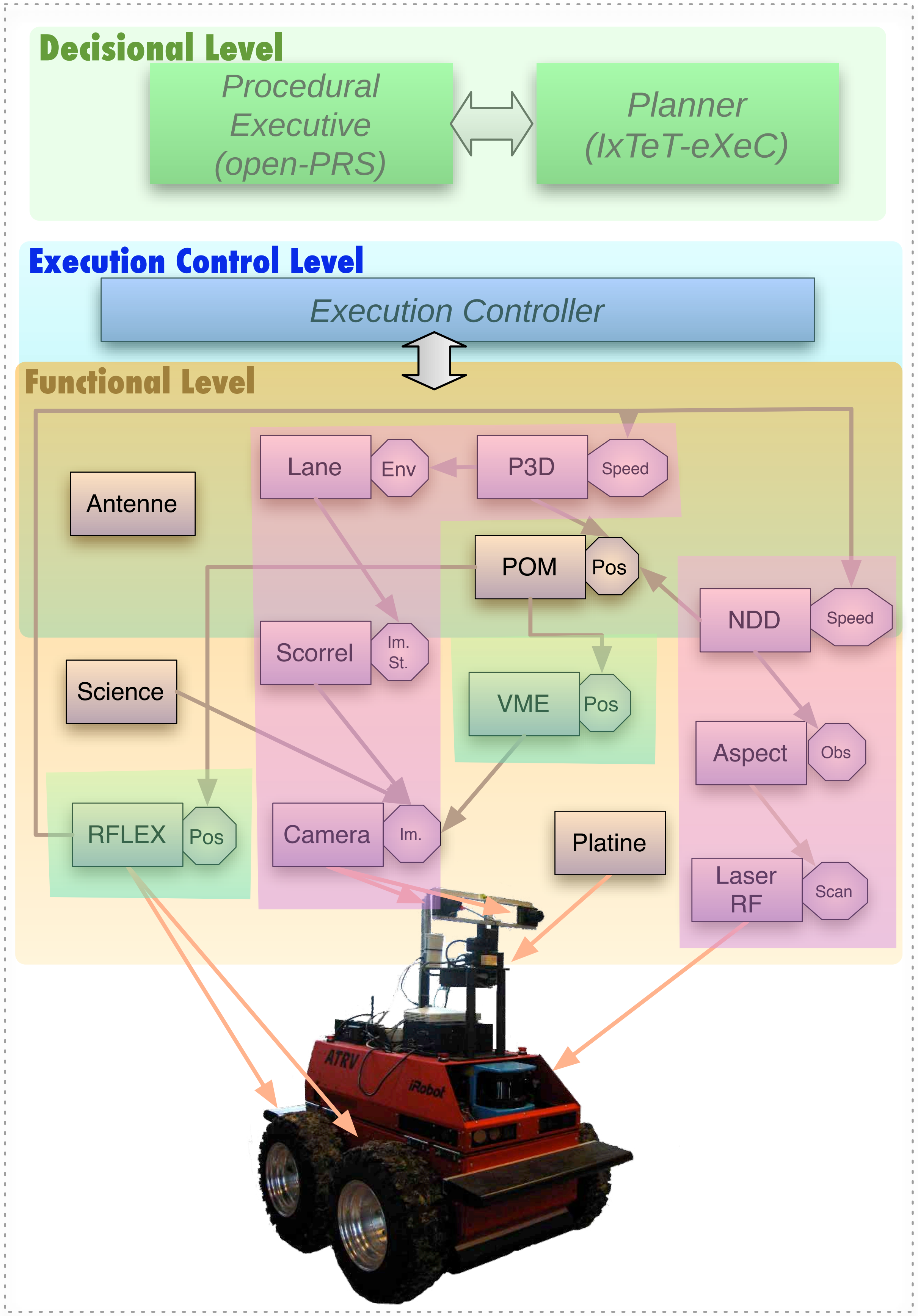


Functional Level

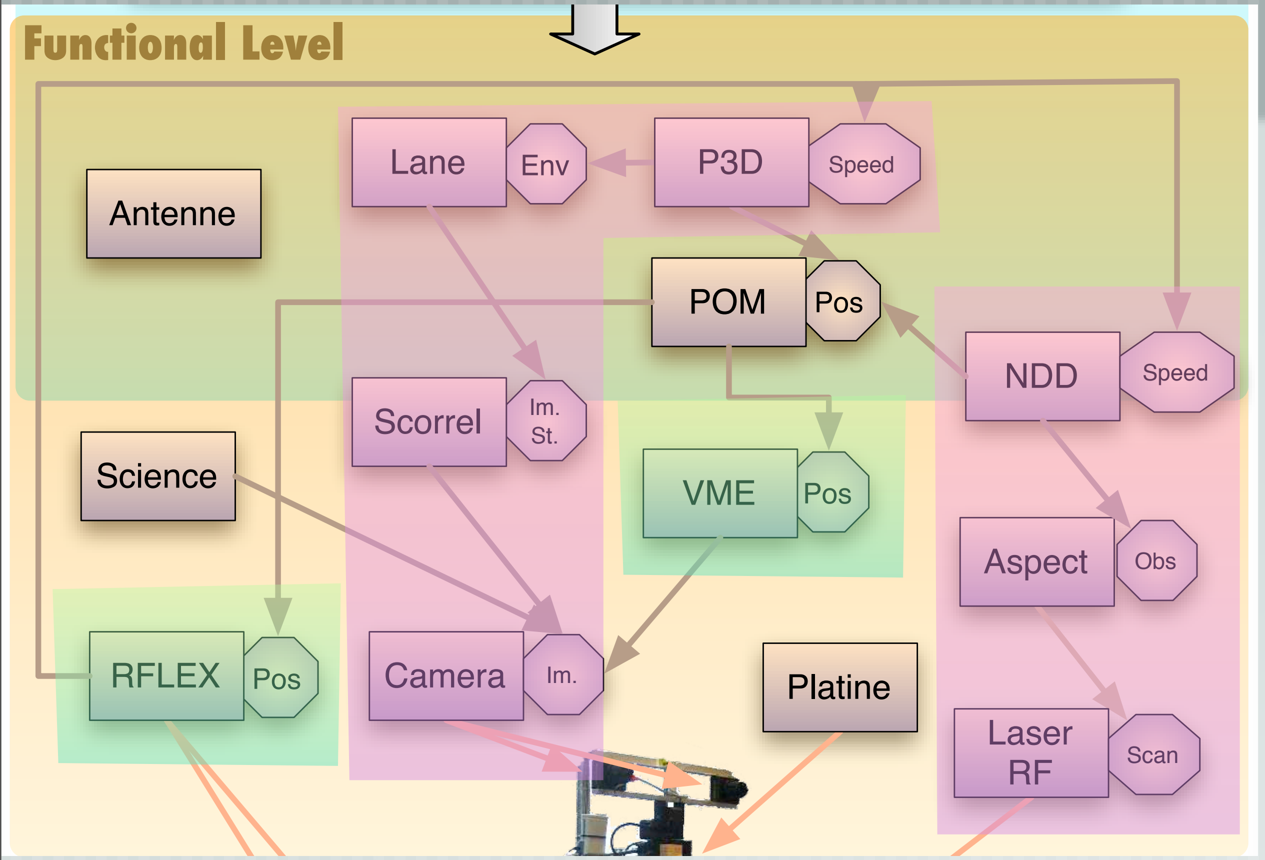




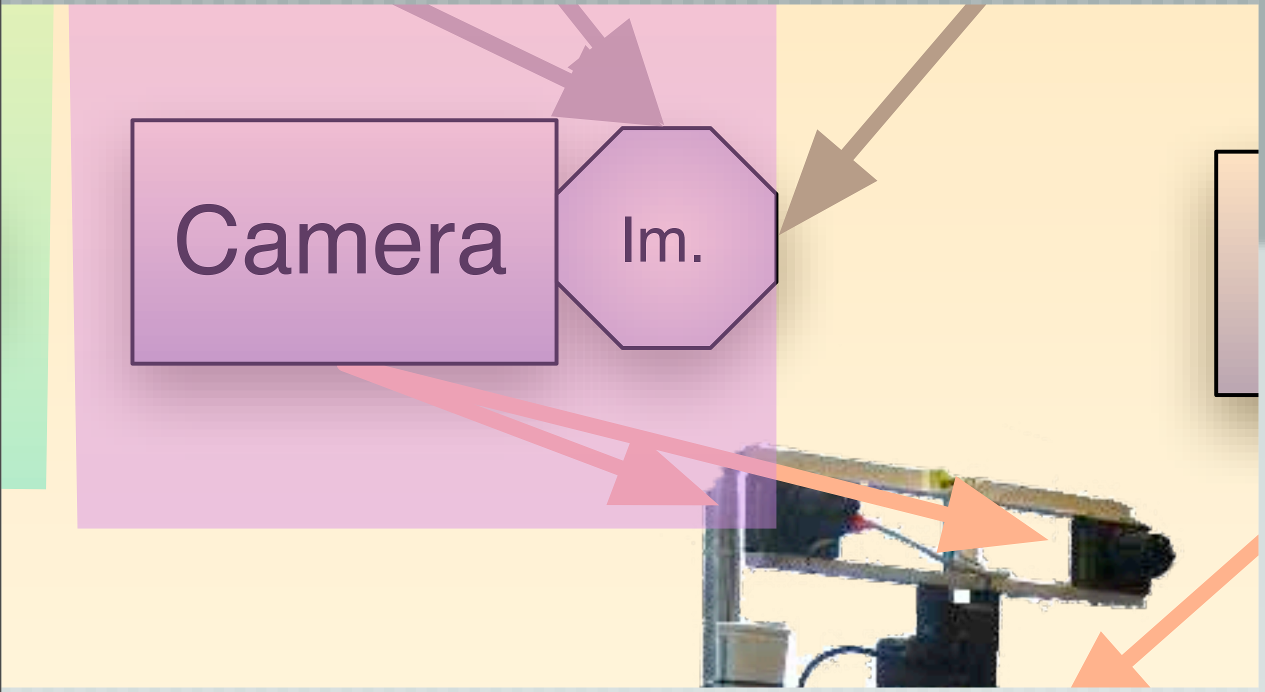
GenoM


Decisional Level
Procedural Executive (open-PRS)
Planner (IxTeT-eXeC)
Execution Control Level
Execution Controller
Functional Level
Lane
Env
P3D
Speed
Antenne
POM
Pos
NDD
Speed
Scorrel
Im. St.
Science
VME
Pos
Aspect
Obs
RFLEX
Pos
Camera
Im.
Platine
Laser RF
Scan
ATRV
iRobot

GenoM

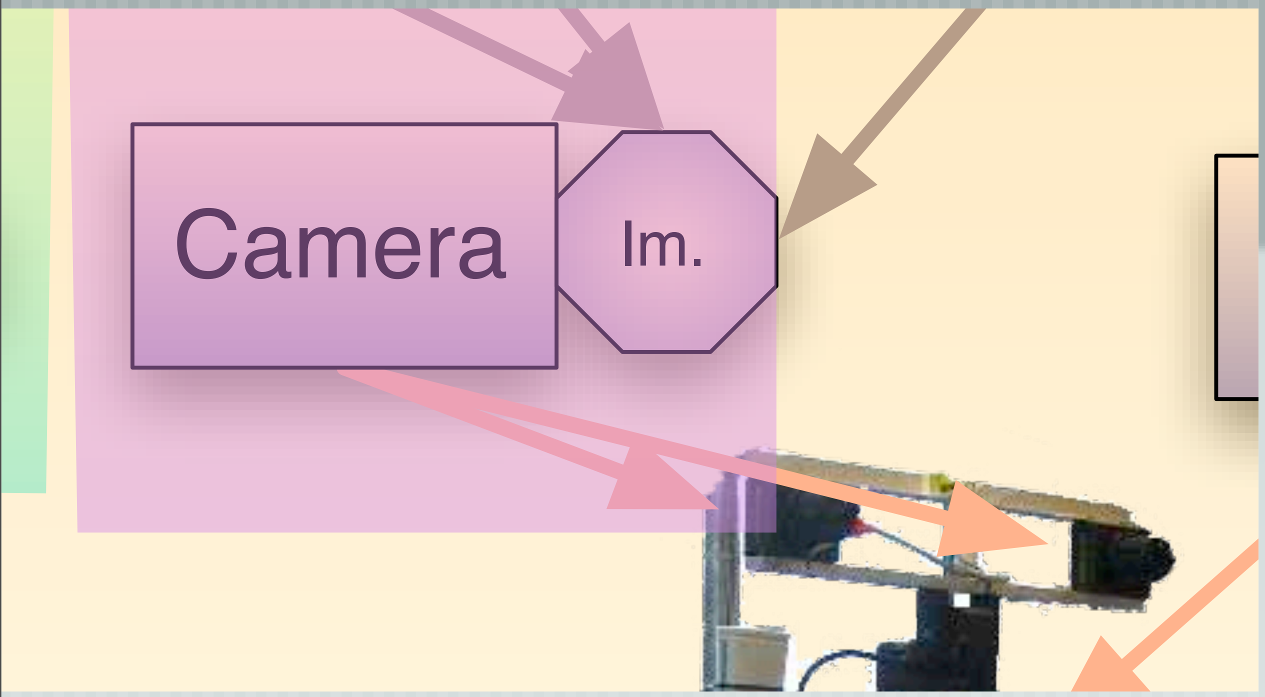

Functional Level
Antenne
Lane
Env
P3D
Speed
POM
Pos
NDD
Speed
Scorrel
Im.
St.
Science
VME
Pos
Aspect
Obs
RFLEX
Pos
Camera
Im.
Platine
Laser
RF
Scan

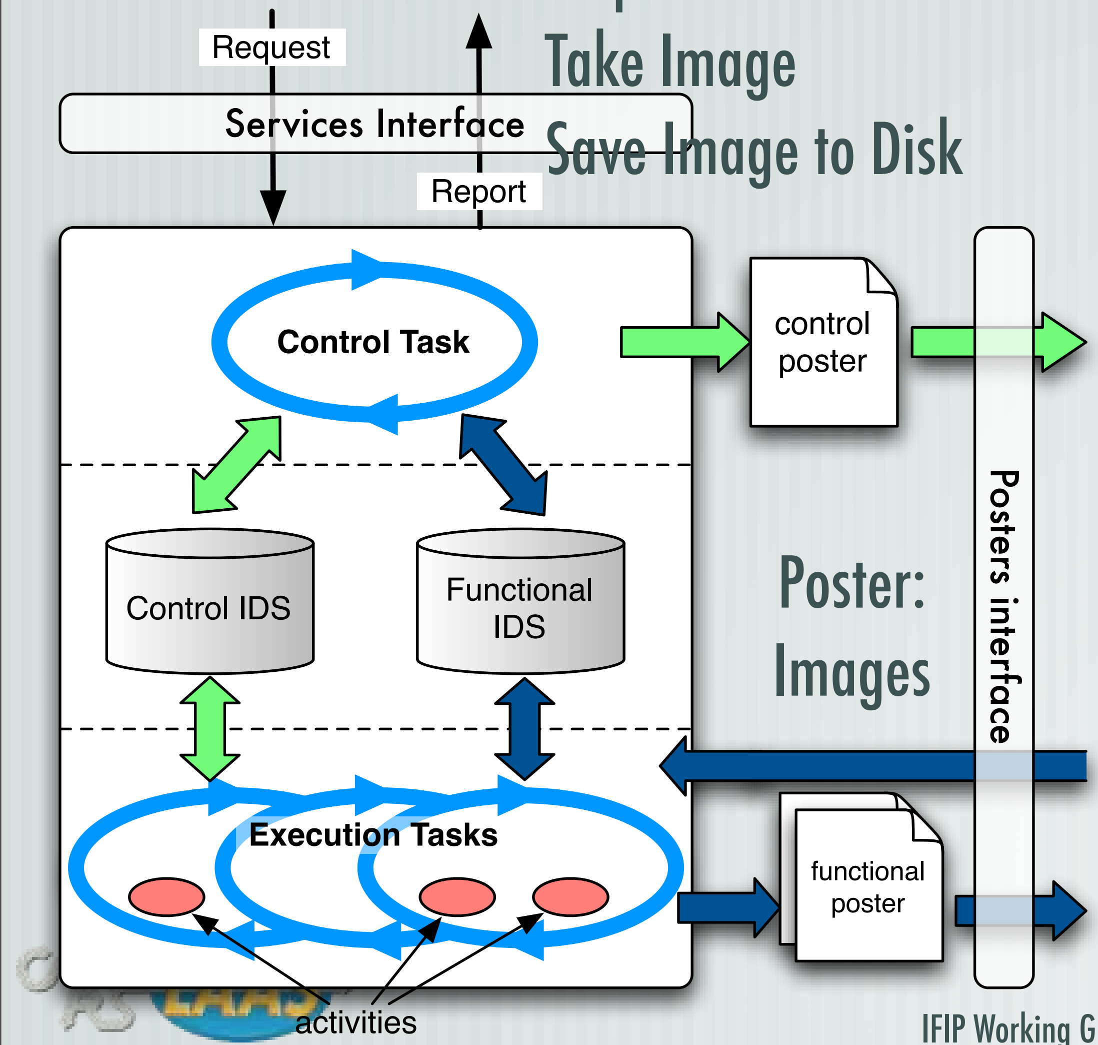
GenoM



GenoM

- Each module is an instance of this one



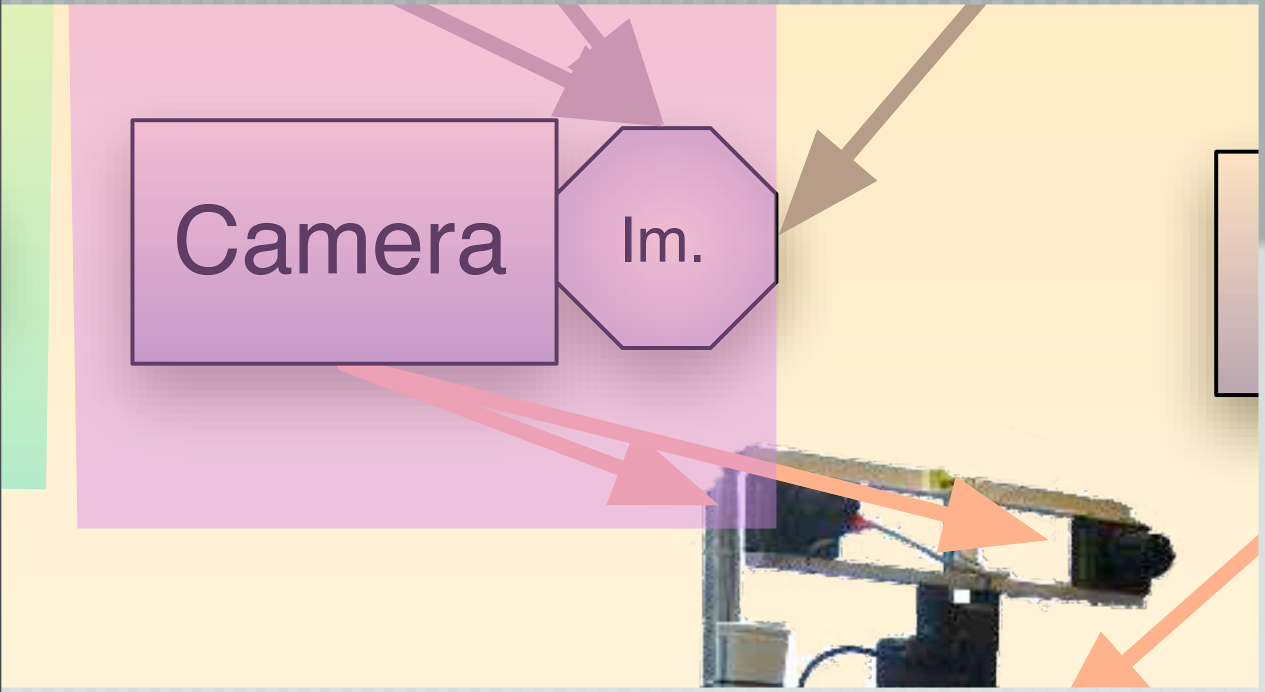


Requests:
Take Image
Save Image to Disk

Poster:
Images

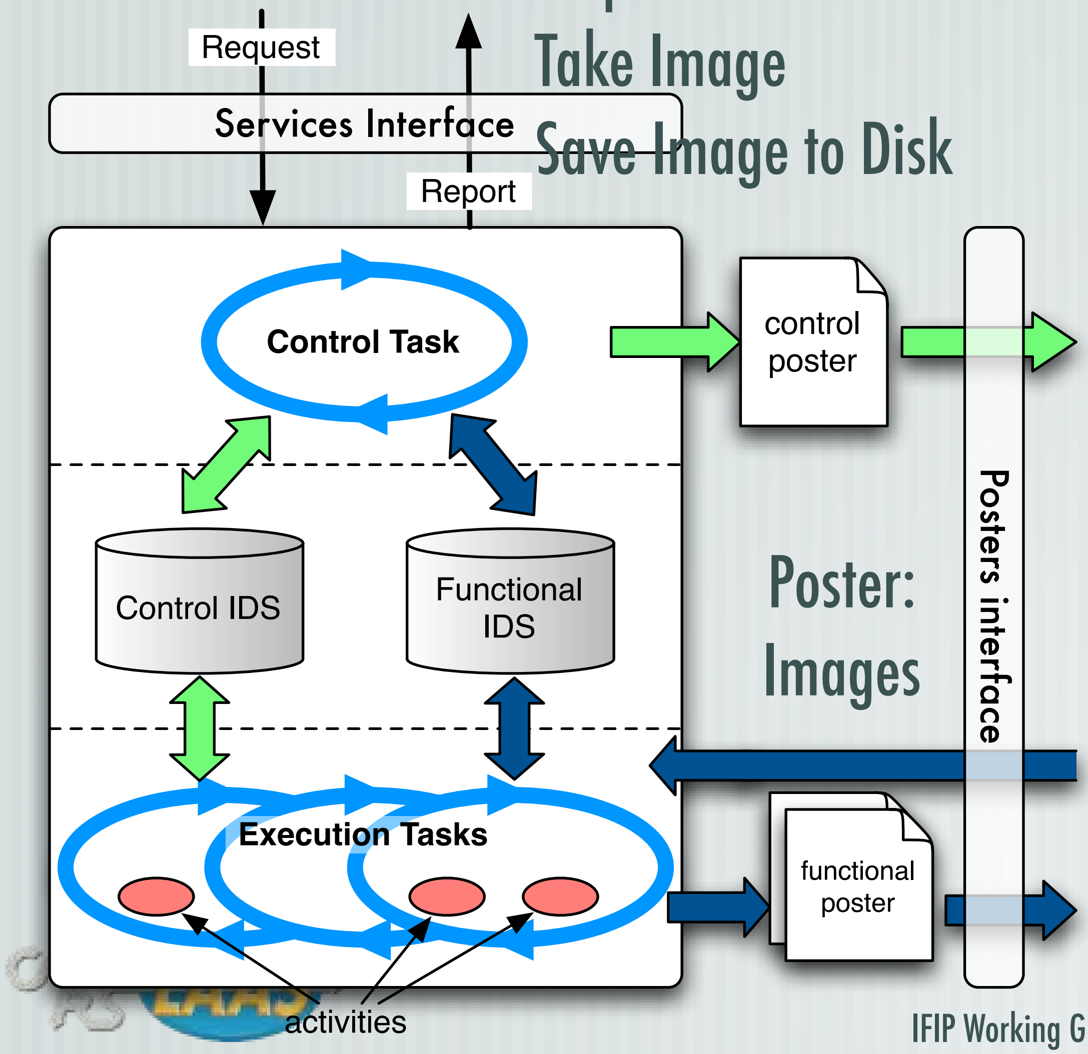
GenoM

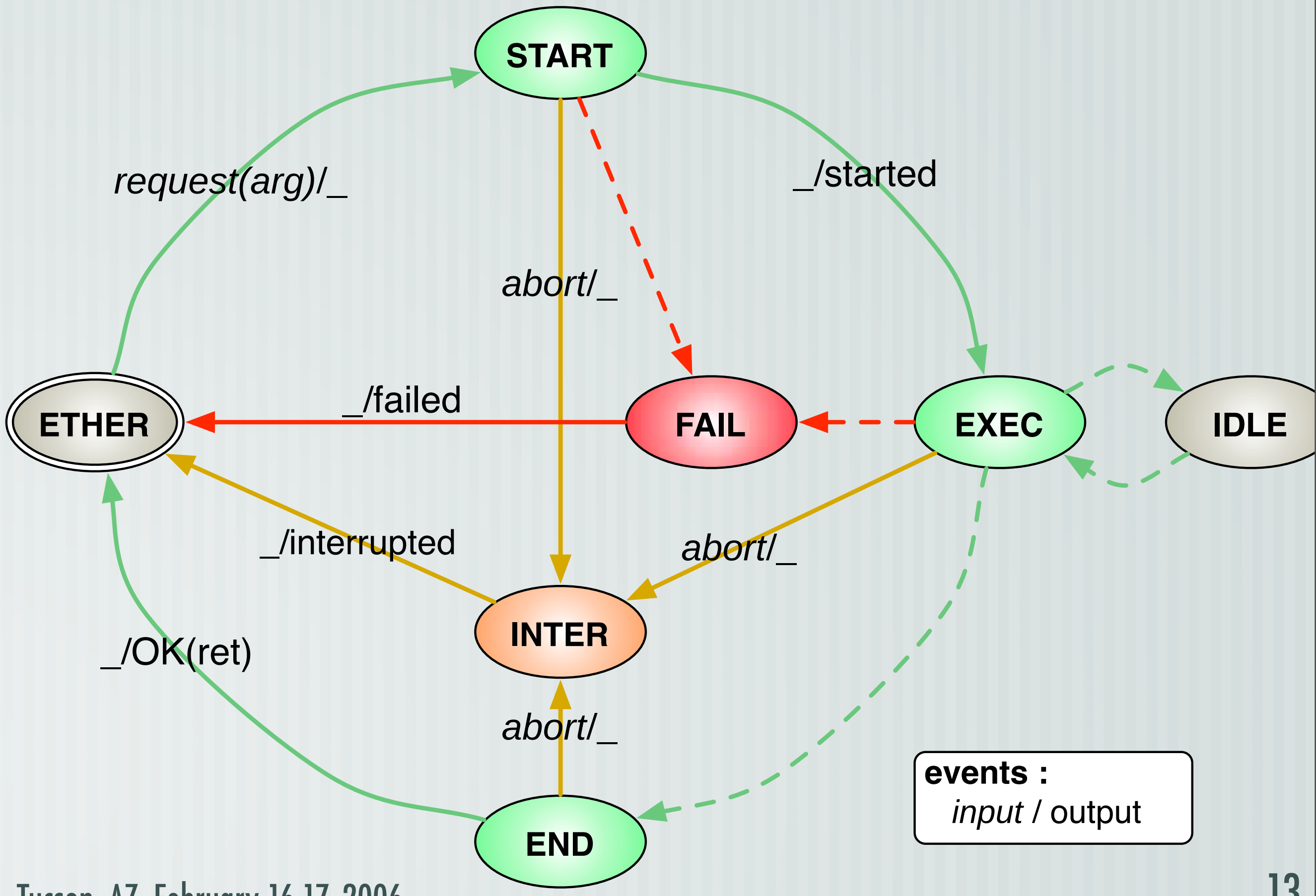
- Each module is an instance of this one
- Each activity runs an automaton such as this one



Requests:
Take Image
Save Image to Disk

Poster:
Images



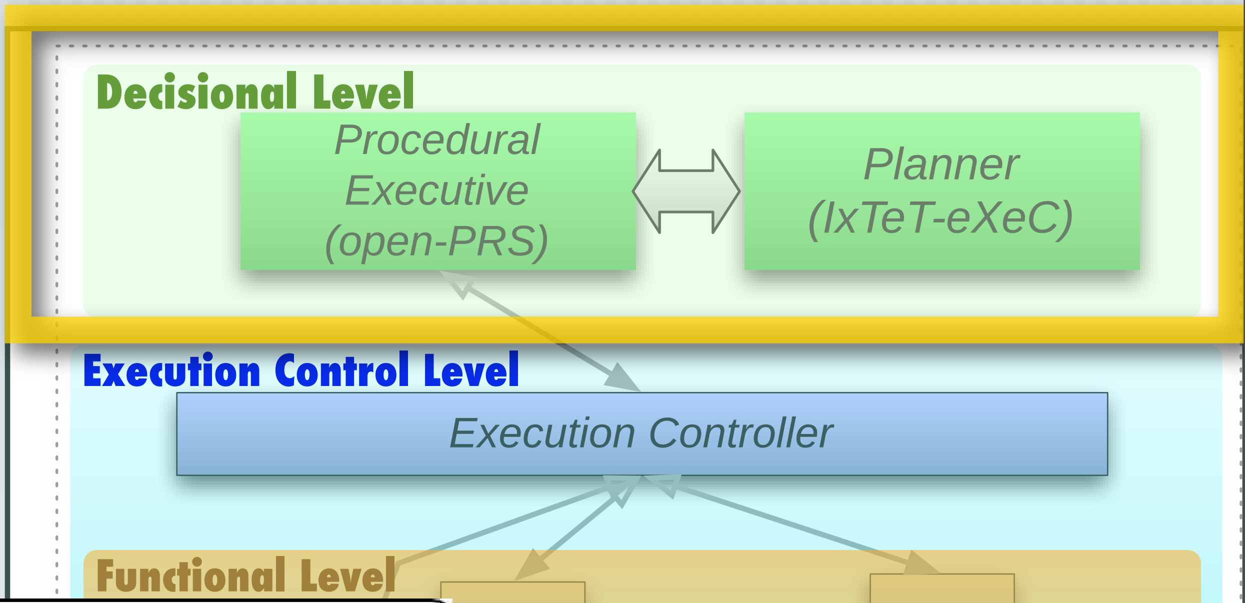


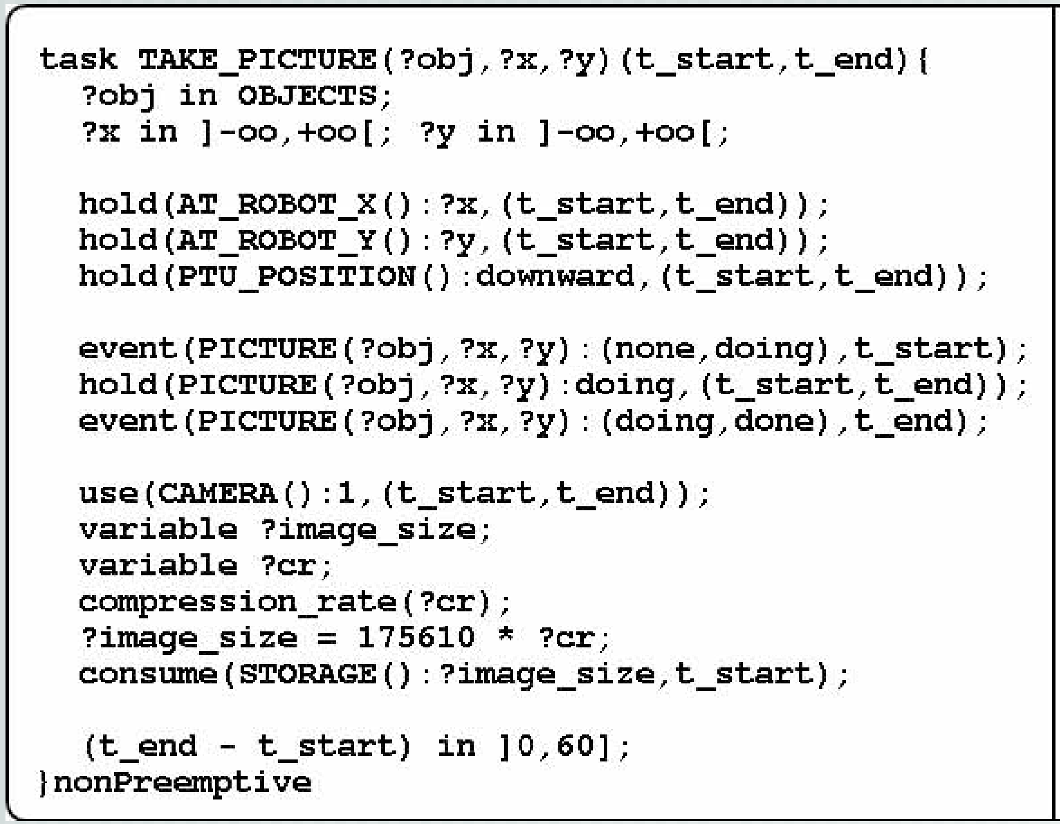
GenoM

- Provides a “software engineering” framework
- Many implementation aspects are relieved from the programmer (communication, threading, etc)
- Internal automaton for the internal activities

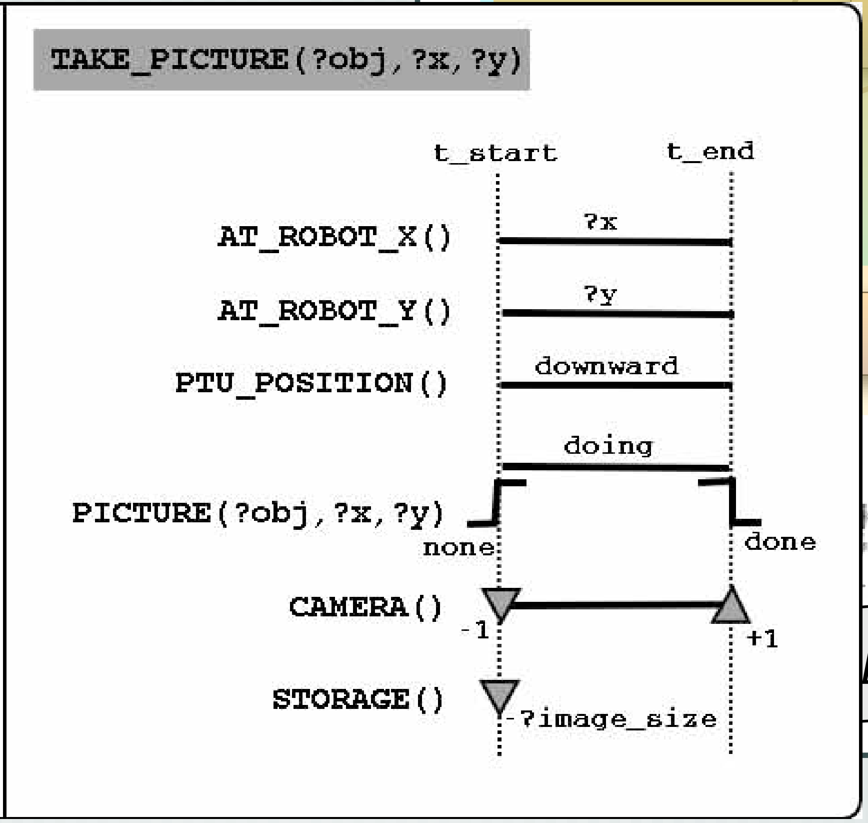
Decisional Level (Task Planning)

- IxTeT
  - Action representation
  - Given a goal and a state produce a plan to reach it
  - Repair and replan



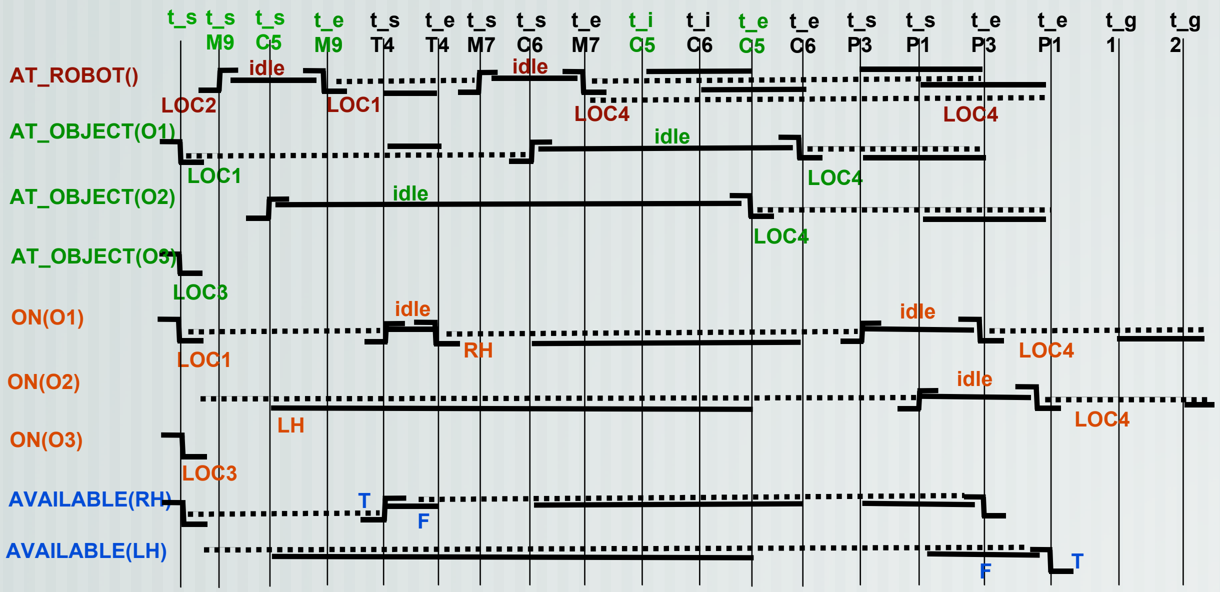


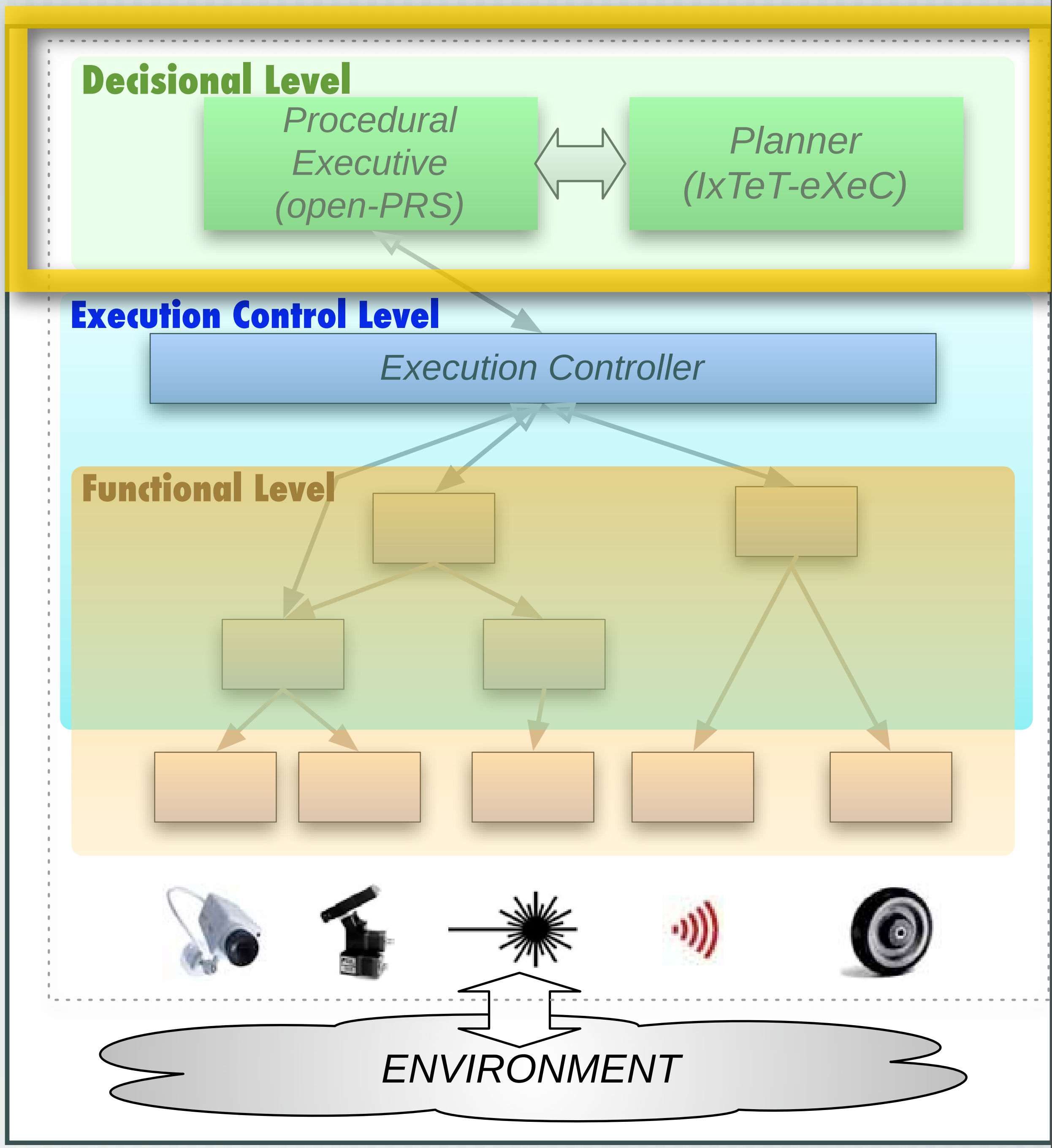
```
task TAKE_PICTURE(?obj,?x,?y)(t_start,t_end){
  ?obj in OBJECTS;
  ?x in ]-oo,+oo[; ?y in ]-oo,+oo[;

  hold(AT_ROBOT_X():?x,(t_start,t_end));
  hold(AT_ROBOT_Y():?y,(t_start,t_end));
  hold(PTU_POSITION():downward,(t_start,t_end));

  event(PICTURE(?obj,?x,?y):(none,doing),t_start);
  hold(PICTURE(?obj,?x,?y):doing,(t_start,t_end));
  event(PICTURE(?obj,?x,?y):(doing,done),t_end);

  use(CAMERA():1,(t_start,t_end));
  variable ?image_size;
  variable ?cr;
  compression_rate(?cr);
  ?image_size = 175610 * ?cr;
  consume(STORAGE():?image_size,t_start);

  (t_end - t_start) in ]0,60];
}nonPreemptive
```



Decisional Level (Task Planning)

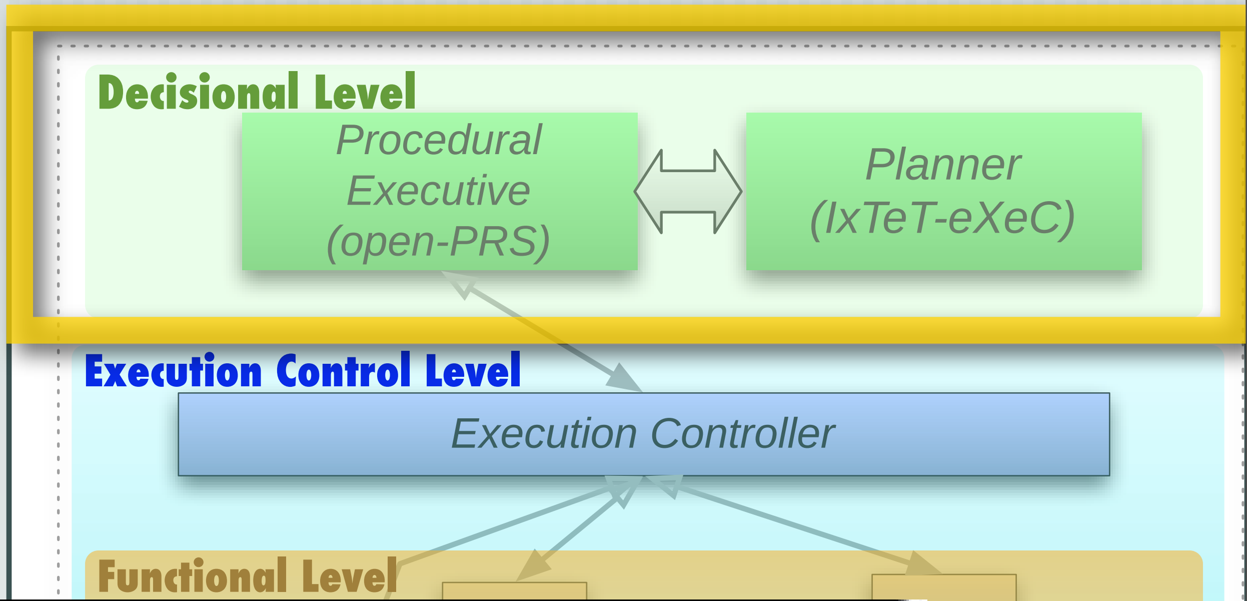
- IxTeT
  - Action representation
  - Given a goal and a state produce a plan to reach it
  - Repair and replan

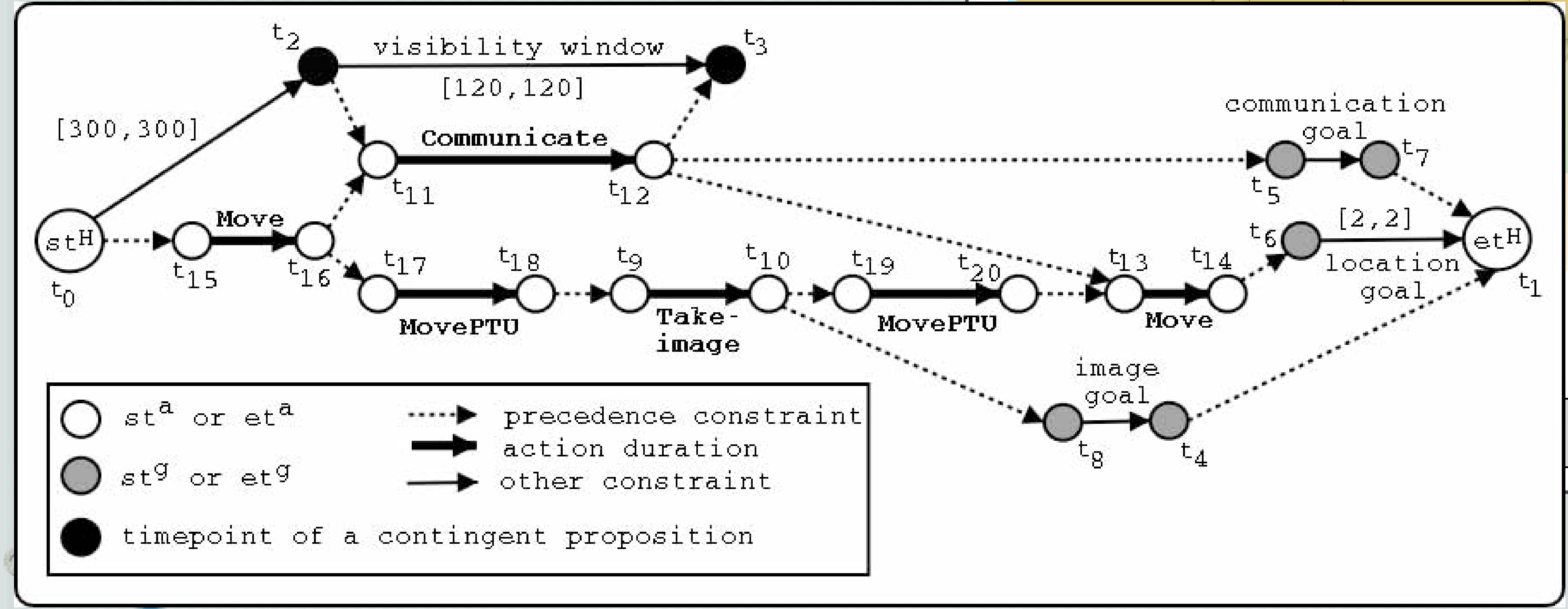




Decisional Level (Task Planning)

- IxTeT
  - Action representation
  - Given a goal and a state produce a plan to reach it
  - Repair and replan





○ st^a or et^a

● st^g or et^g

● timepoint of a contingent proposition

┈▶ precedence constraint

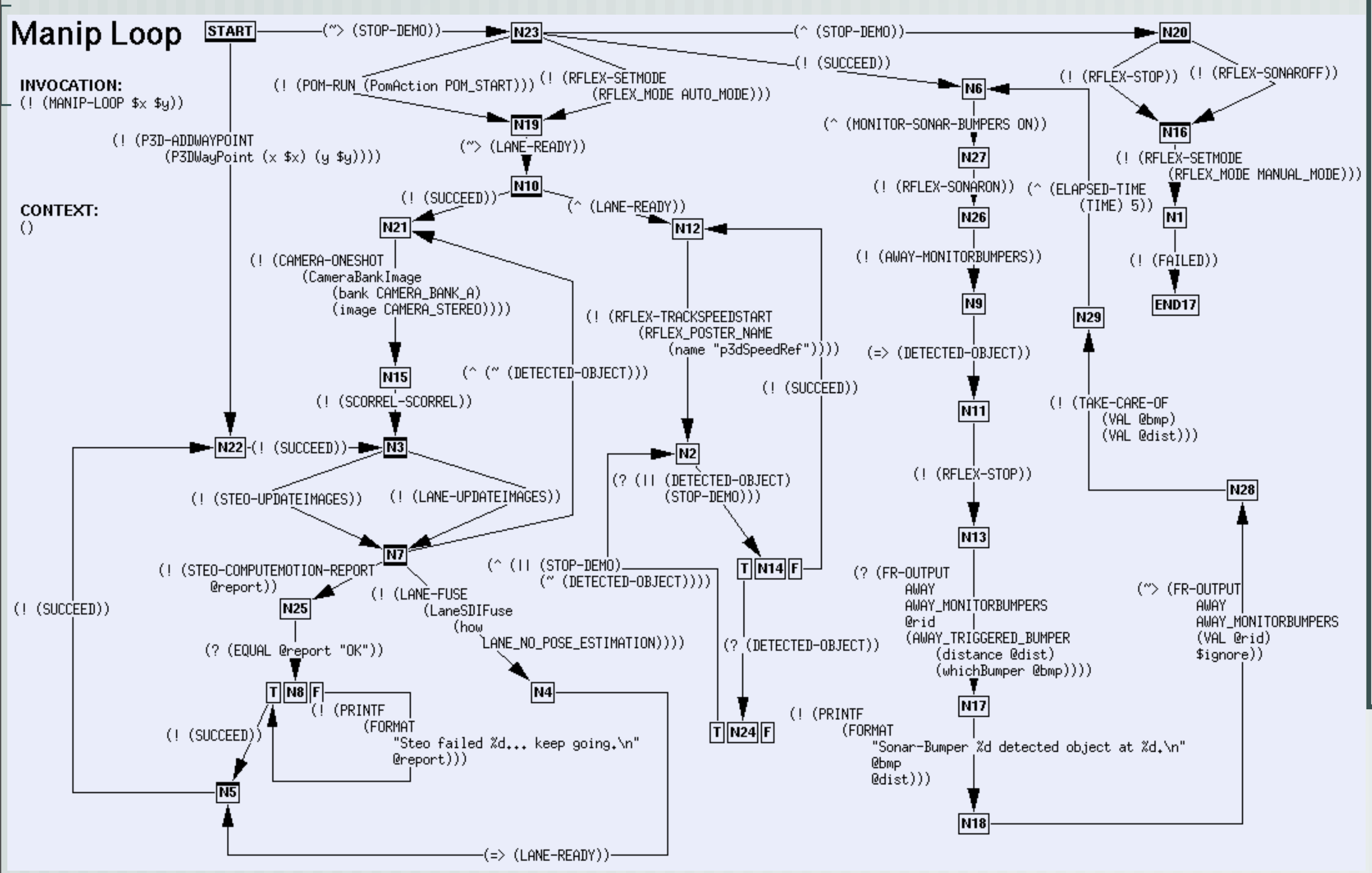
━▶ action duration

─▶ other constraint

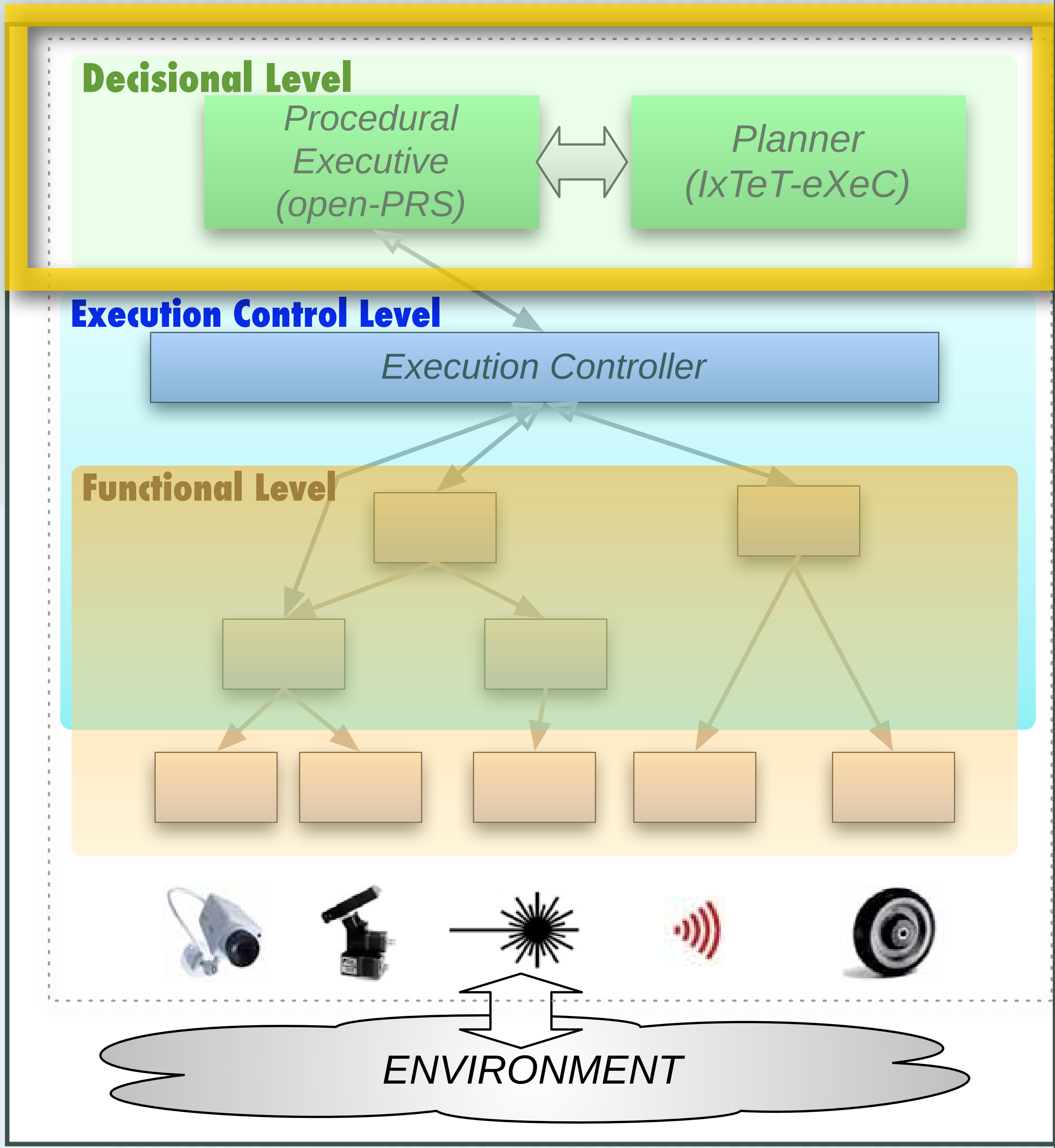
Decisional Level (Procedural Executive)

OpenPRS

- Refine high level “task”
- Some local recoveries
- Goal and Data driven procedures
- Use procedural reasoning



Manip Loop
START
INVOCATION:
(! (MANIP-LOOP $x $y))
CONTEXT:
()
(~> (STOP-DEMO))
N23
(^ (STOP-DEMO))
N20
(! (SUCCEED))
(! (POM-RUN (PomAction POM_START)))
(! (RFLEX-SETMODE (RFLEX_MODE AUTO_MODE)))
N6
(! (RFLEX-STOP))
(! (RFLEX-SONAROFF))
N19
(! (P3D-ADDWAYPOINT (P3DWayPoint (x $x) (y $y))))
(~> (LANE-READY))
(^ (MONITOR-SONAR-BUMPERS ON))
N16
N27
(! (RFLEX-SETMODE (RFLEX_MODE MANUAL_MODE)))
N10
(! (SUCCEED))
(^ (LANE-READY))
(! (RFLEX-SONARON))
(^ (ELAPSED-TIME (TIME) 5))
N21
N12
N26
N1
(! (CAMERA-ONESHOT (CameraBankImage (bank CAMERA_BANK_A) (image CAMERA_STEREO))))
(! (AWAY-MONITORBUMPERS))
(! (FAILED))
END17
N9
N29
(! (RFLEX-TRACKSPEEDSTART (RFLEX_POSTER_NAME (name "p3dSpeedRef"))))
(=> (DETECTED-OBJECT))
N15
(^ (~ (DETECTED-OBJECT)))
(! (SUCCEED))
(! (SCORREL-SCORREL))
N11
(! (TAKE-CARE-OF (VAL @bmp) (VAL @dist)))
N22
(! (SUCCEED))
N3
N2
(? (|| (DETECTED-OBJECT) (STOP-DEMO)))
(! (RFLEX-STOP))
N28
(! (STEO-UPDATEIMAGES))
(! (LANE-UPDATEIMAGES))
N7
N13
(^ (|| (STOP-DEMO) (~ (DETECTED-OBJECT))))
T N14 F
(! (STEO-COMPUTEMOTION-REPORT @report))
(? (FR-OUTPUT AWAY_MONITORBUMPERS @rid (AWAY_TRIGGERED_BUMPER (distance @dist) (whichBumper @bmp))))
(~> (FR-OUTPUT AWAY_MONITORBUMPERS (VAL @rid) $ignore))
(! (SUCCEED))
N25
(! (LANE-FUSE (LaneSDIFuse (how LANE_NO_POSE_ESTIMATION))))
(? (EQUAL @report "OK"))
(? (DETECTED-OBJECT))
N4
T N8 F
(! (PRINTF (FORMAT "Steo failed %d... keep going.\n" @report)))
N17
(! (SUCCEED))
T N24 F
(! (PRINTF (FORMAT "Sonar-Bumper %d detected object at %d.\n" @bmp @dist)))
N5
N18
(=> (LANE-READY))



Decisional Level

- Brings some operational “robustness”
  - Plan repair
  - Failure recovery from the Procedural Executive

Dependability of Autonomous Systems

- Functional level hard to validate :
  - we may validate 1 module (synchronous language, UPPAAL, Spin ...)
  - but hard to validate tens and their concurrent interactions...
- Decisional based on AI concept (complex formalism, ...)
- Environment can hardly be modeled (unforeseen evolutions, ...)

Dependability of Autonomous Systems

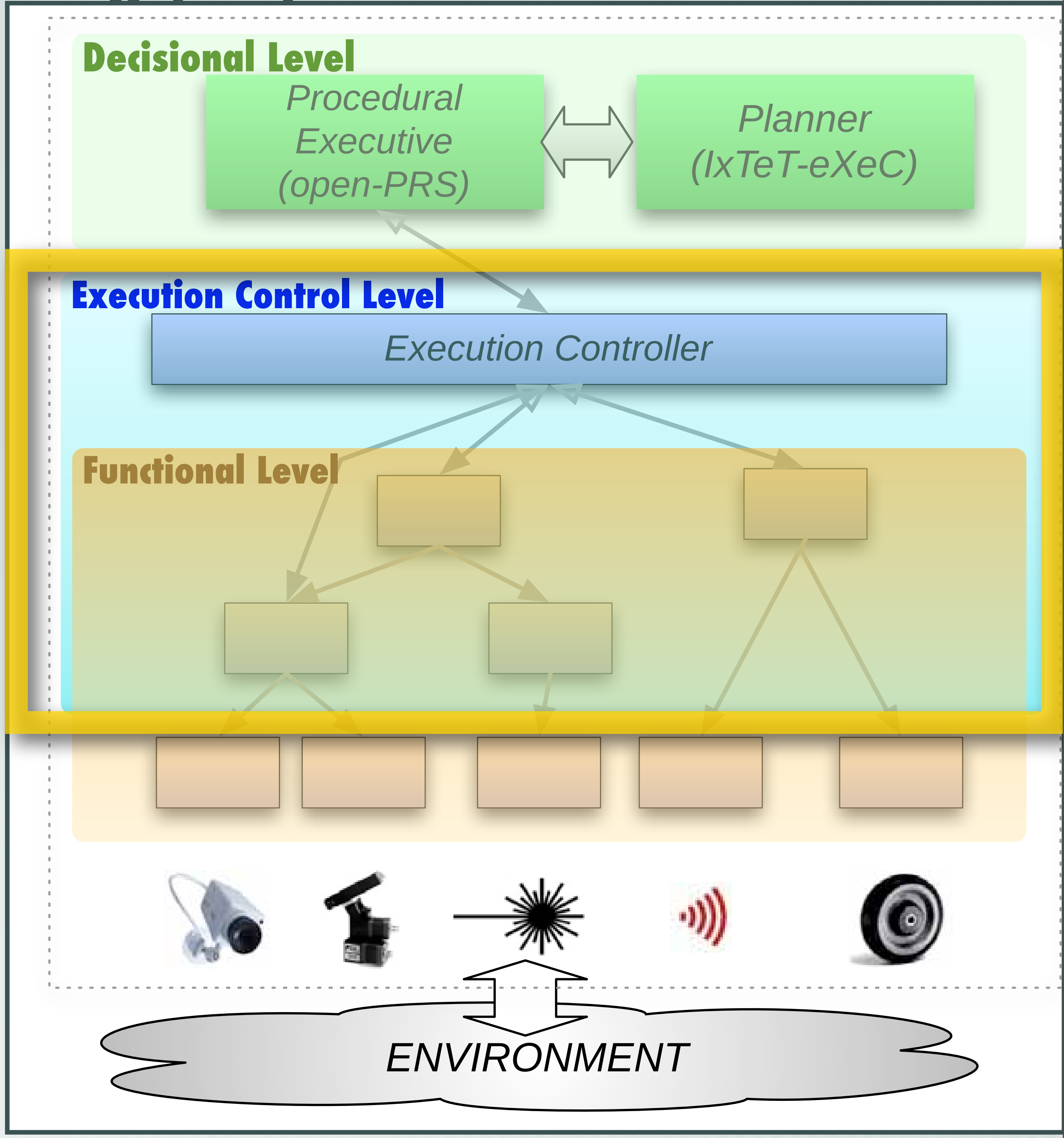
- Functional level hard to validate :
  - we may validate 1 module (synchronous language, UPPAAL, Spin ...)
  - but hard to validate tens and their concurrent interactions...
- Decisional based on AI concept (complex formalism, ...)
- Environment can hardly be modeled (unforeseen evolutions, ...)

Such a system offers little guarantee w.r.t reliability and safety

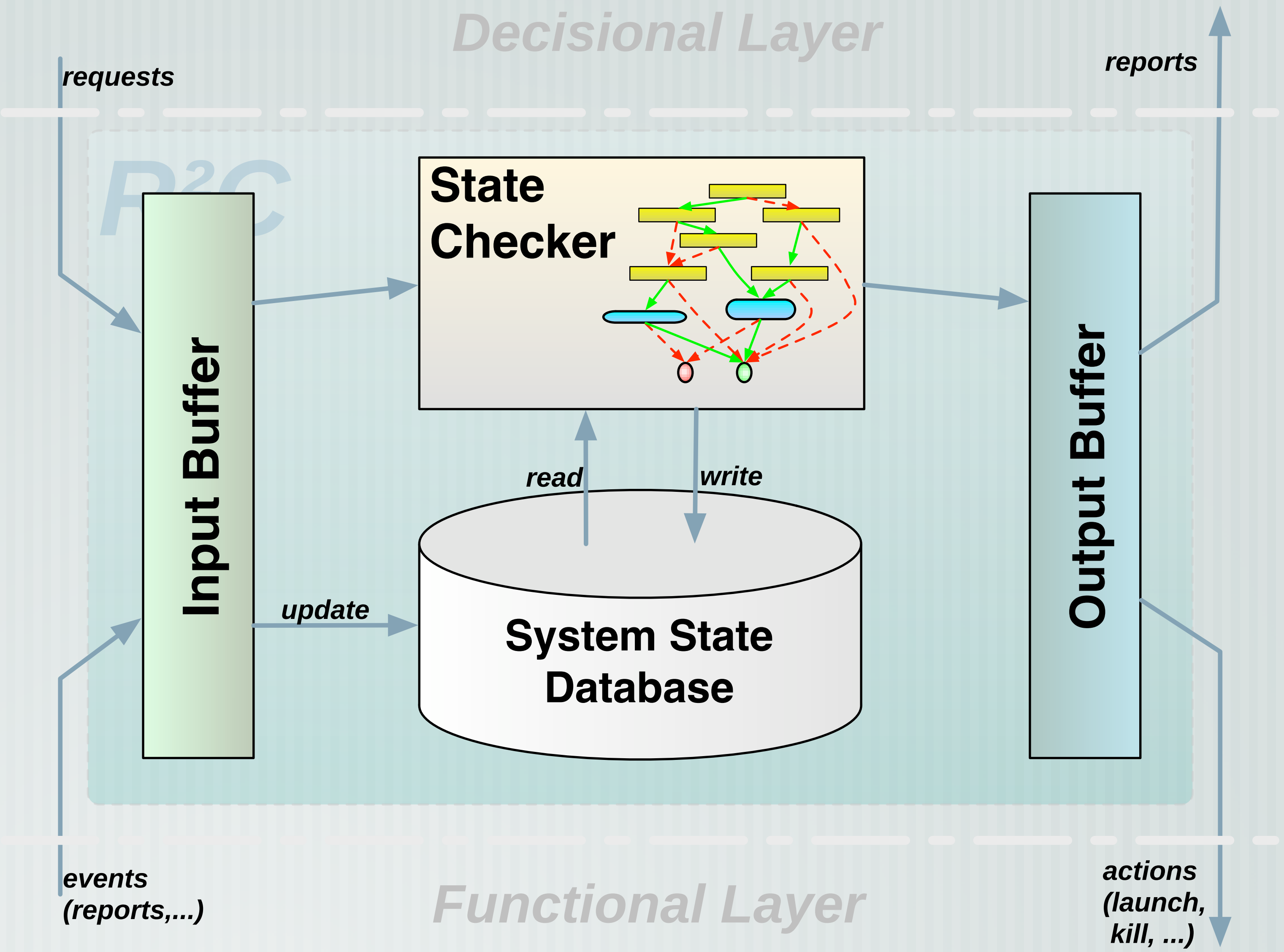
Proposed Solution

The component must have the following properties

Observable	knowledge at all time of the events which may change the state of the system.
Control	ability to act upon these events to maintain the system in a safe and consistent state.
Real Time	decide and act in real-time.
Validation	use formal method.
Simple	ease of use to program the constraints and the rules.
Integrated	well integrated in the rest of the architecture.

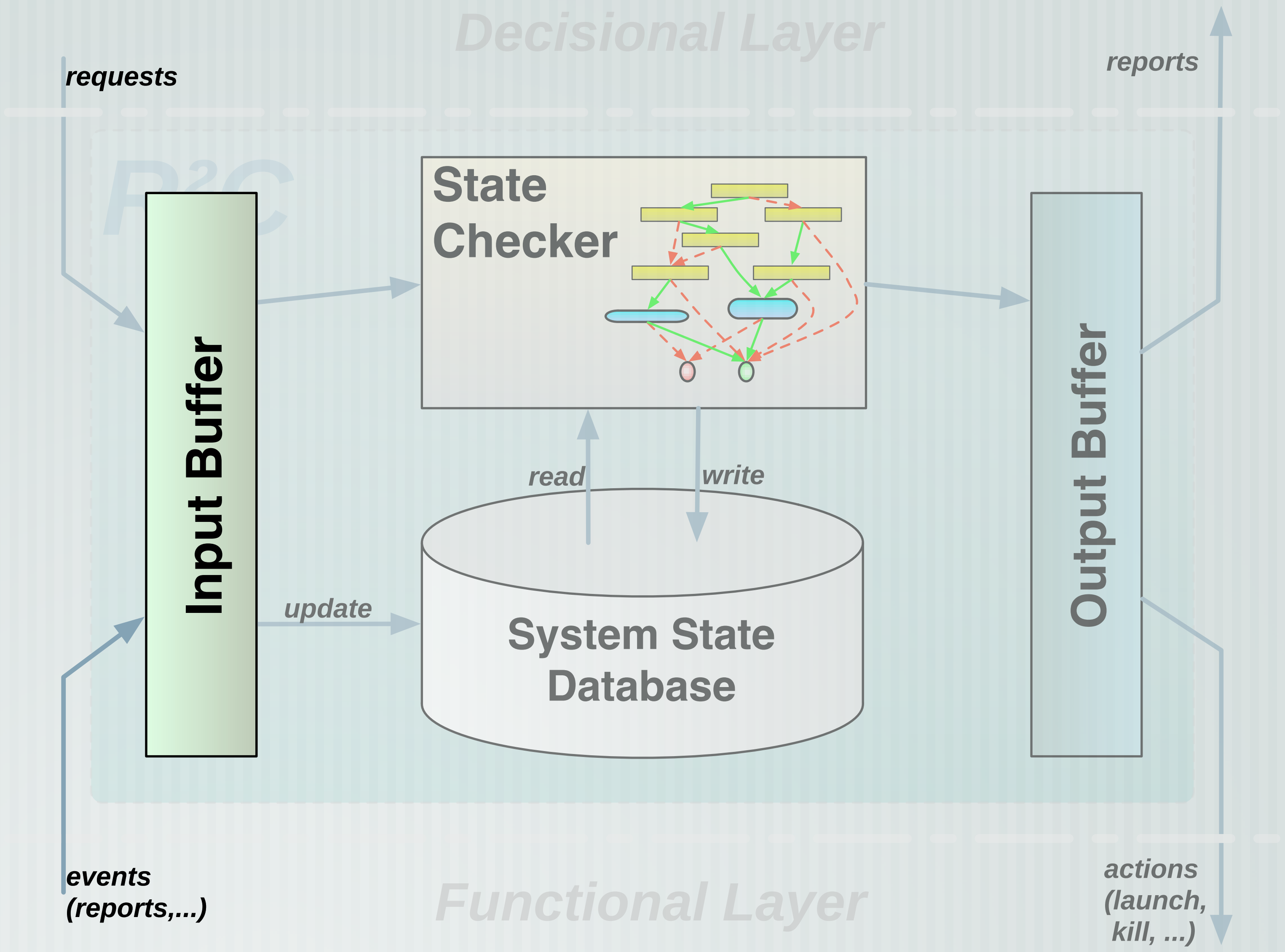


The Request and Report Checker



The Request and Report Checker

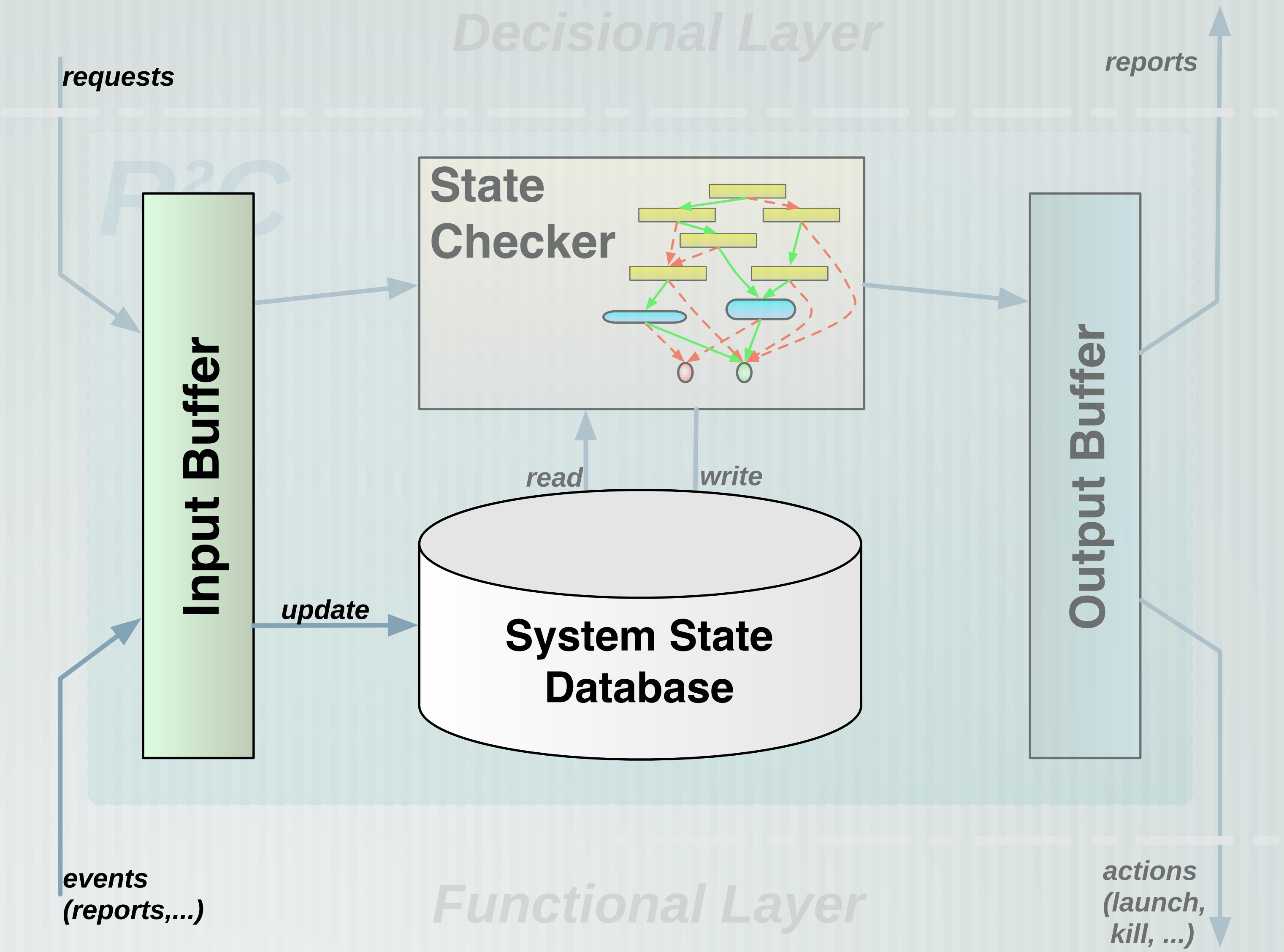
1 Events Capture



The Request and Report Checker

1 Events Capture

2 State Update

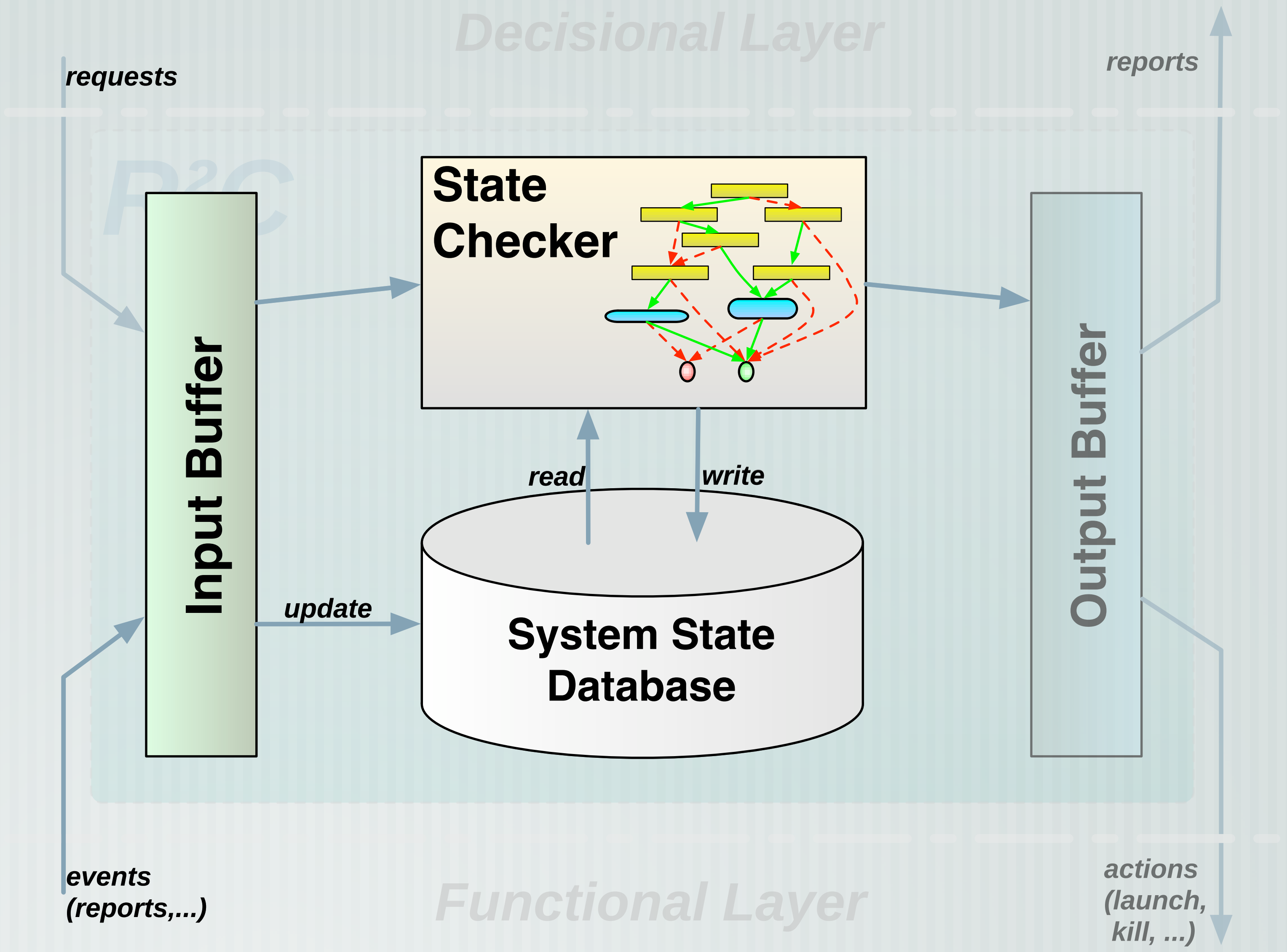


The Request and Report Checker

1 Events Capture

2 State Update

3 State Checker



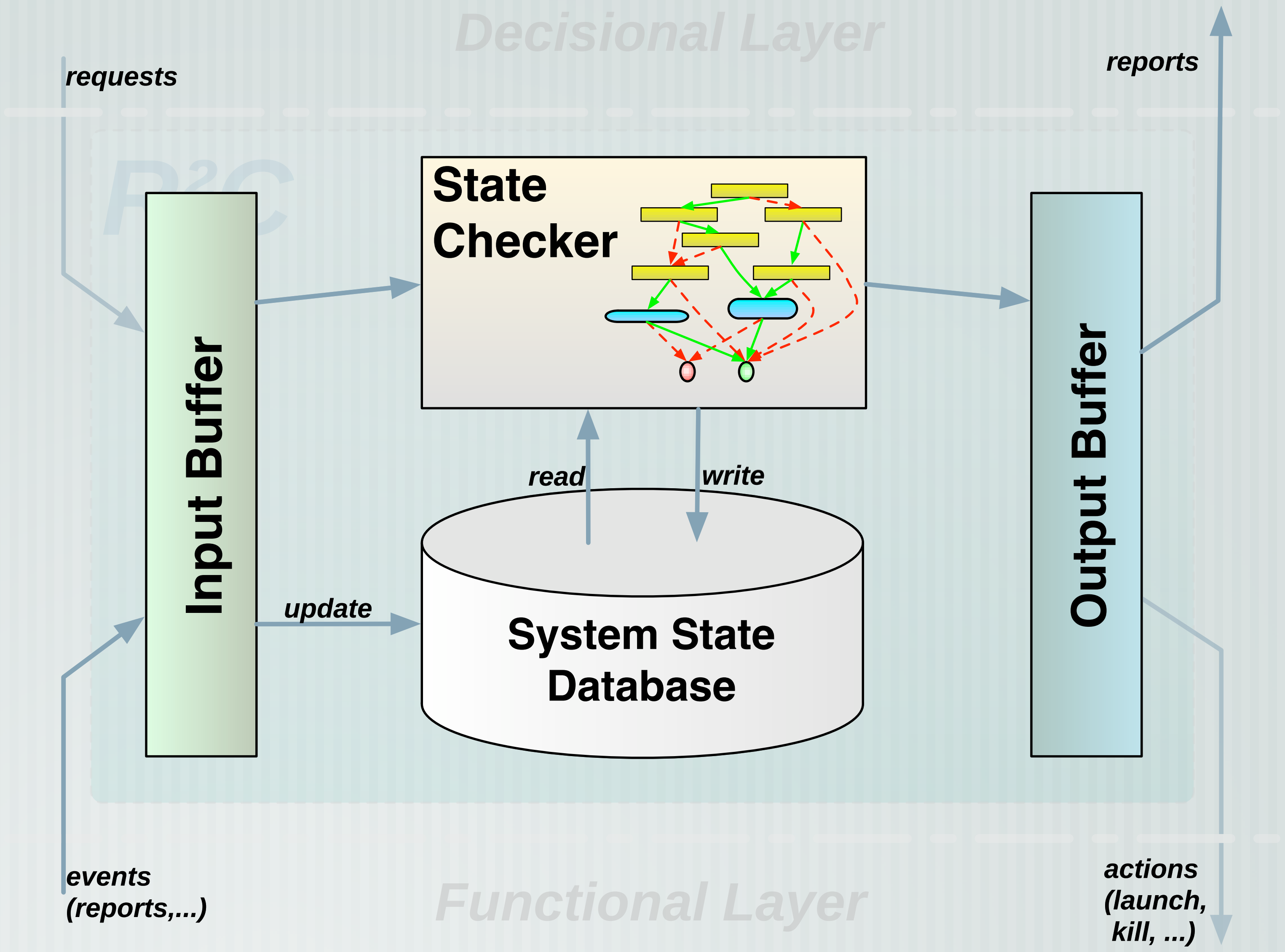
The Request and Report Checker

1 Events Capture

2 State Update

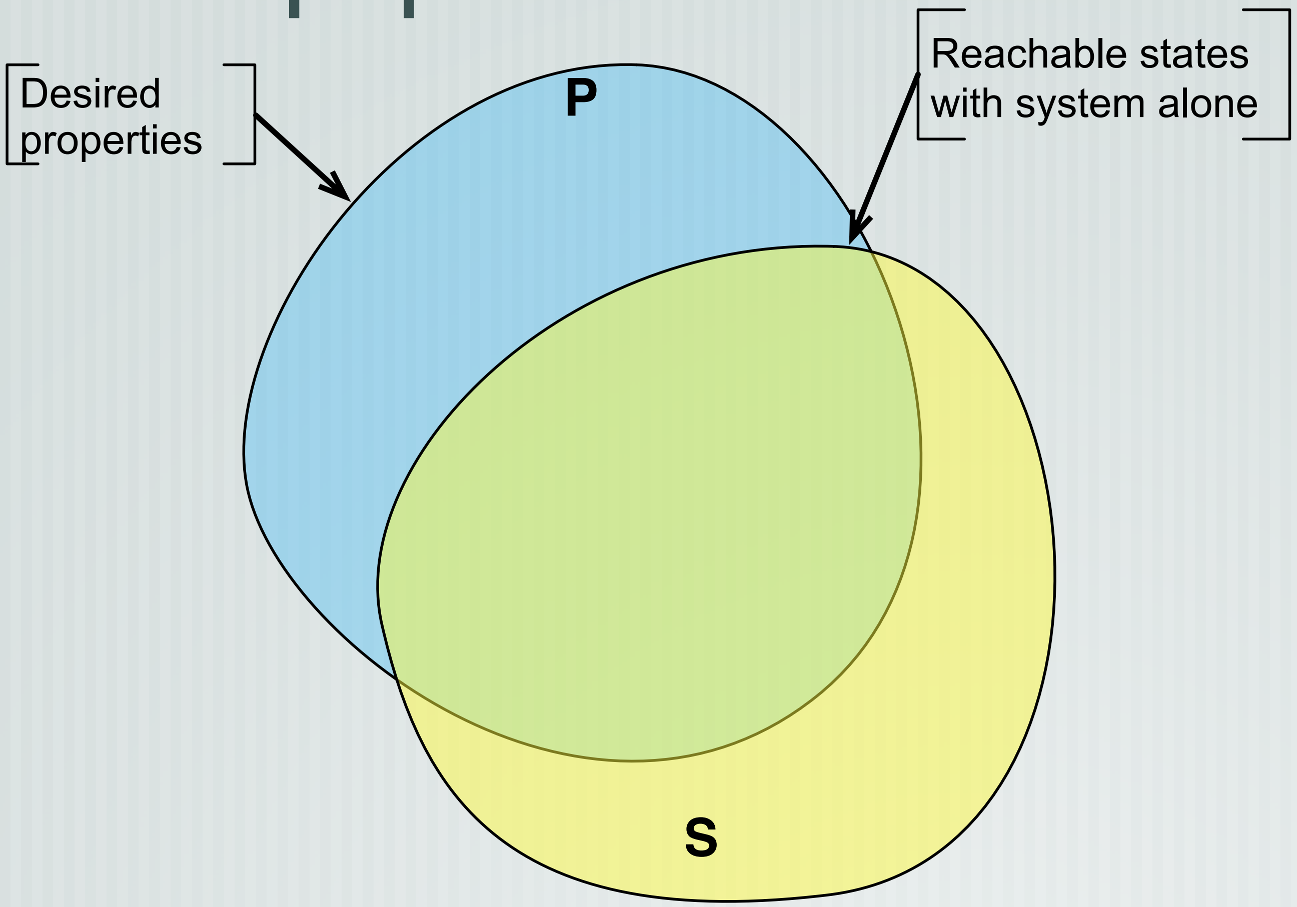
3 State Checker

4 Actions Exec.



Principle

- Formal model of the system (automates, RdP, ...)
- Desired properties



Desired properties

Reachable states with system alone

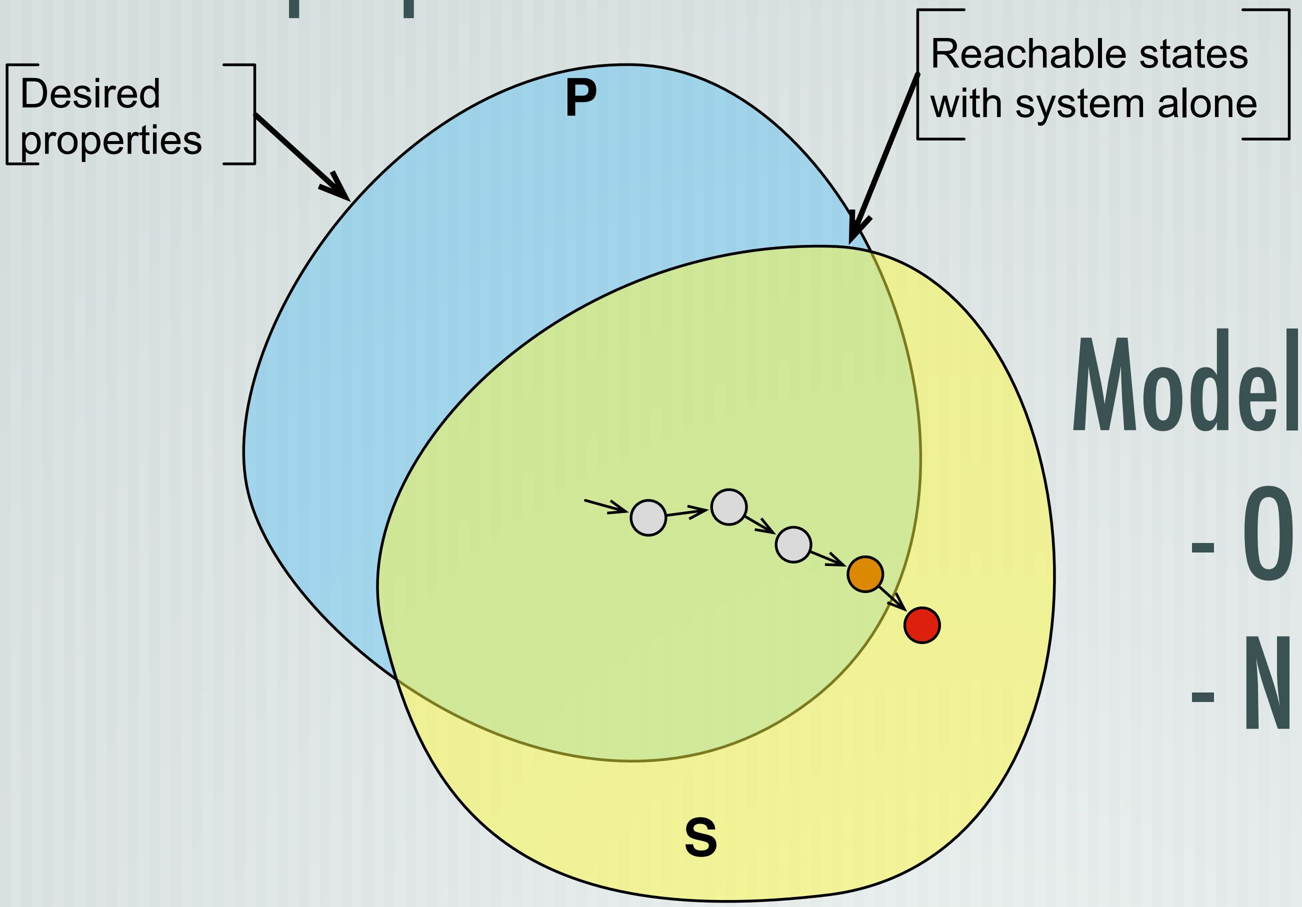
○ Valid State

● Invalid State

● Invalid Successor(s)

Principle

- Formal model of the system (automates, RdP, ...)
- Desired properties



Model-checking :

- OK
- NO + counter example(s)

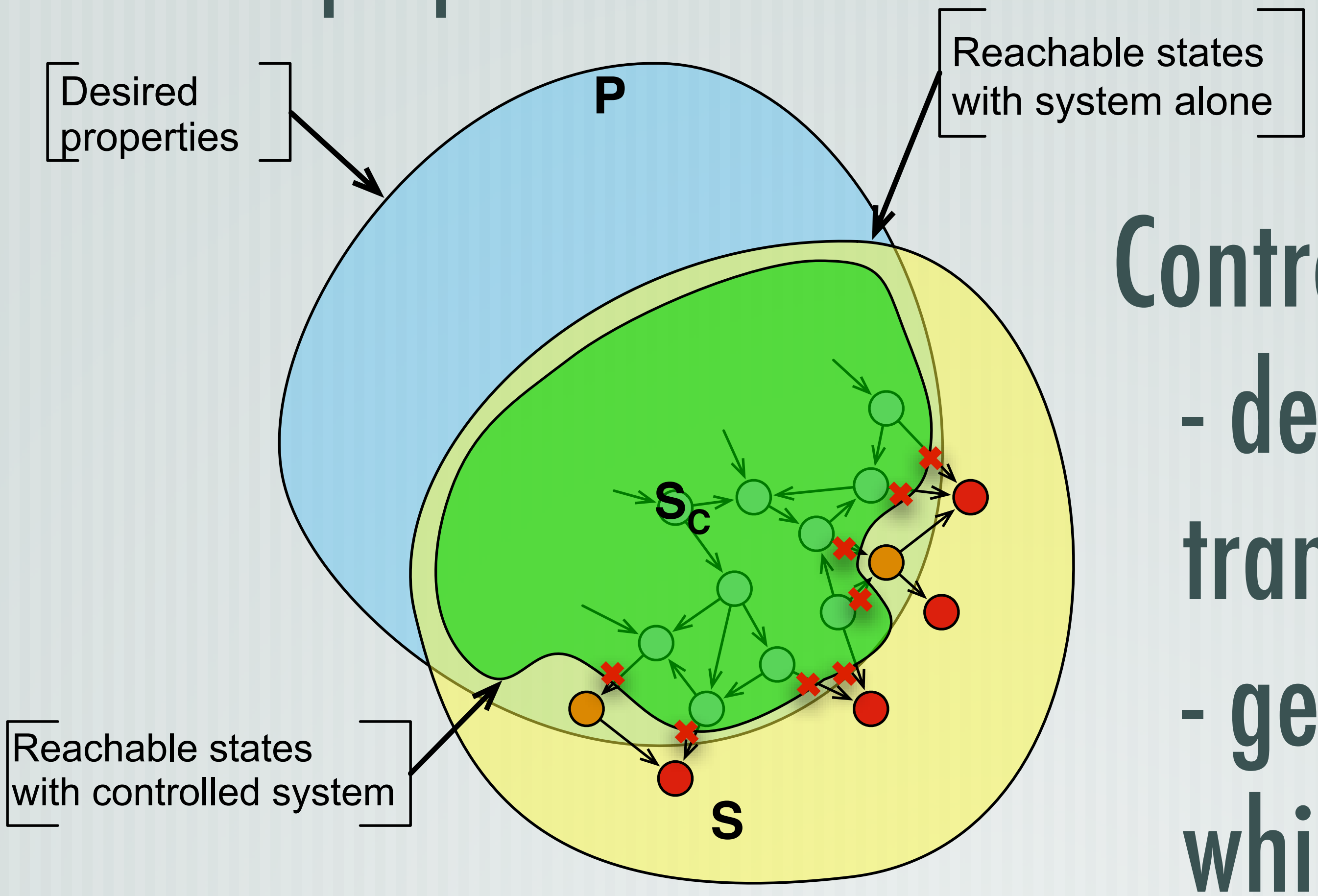
○ Valid State

● Invalid State

● Invalid Successor(s)

Principle

- Formal model of the system (automates, RdP, ...)
- Desired properties


Desired properties
P
Reachable states with system alone
S
c
Reachable states with controlled system
S

Controller synthesis

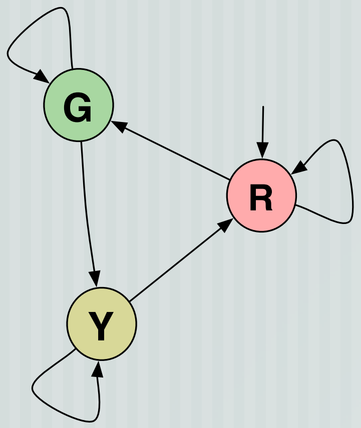
- detect dangerous transitions w.r.t P
- generate a component which blocks them

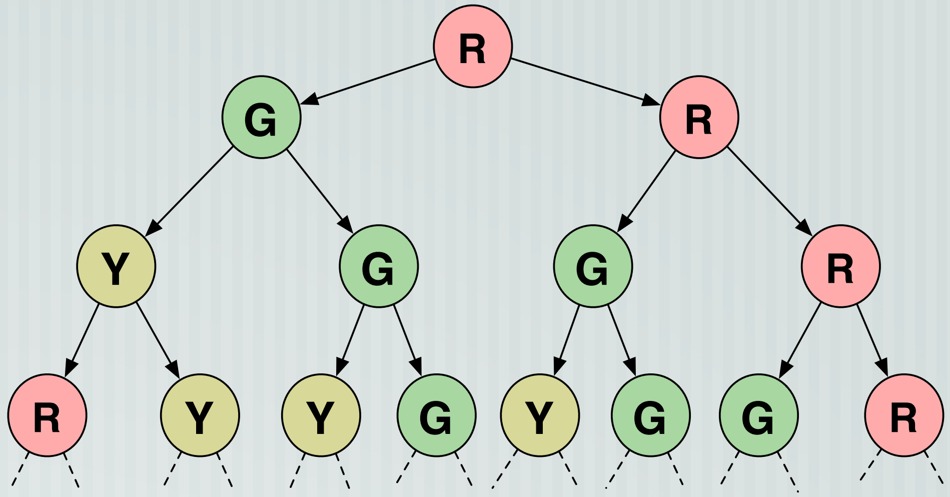
○ **Valid State**
● **Invalid State**
● **Invalid Successor(s)**

Expression of the properties : CTL

CTL : Computational Tree Logic

- time is seen as the tree of possible future



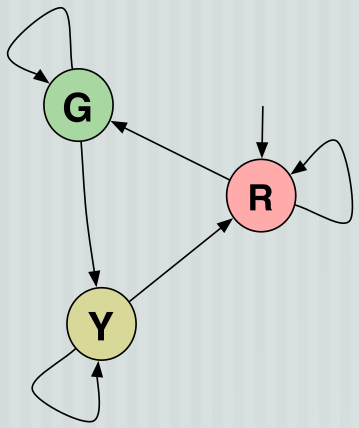


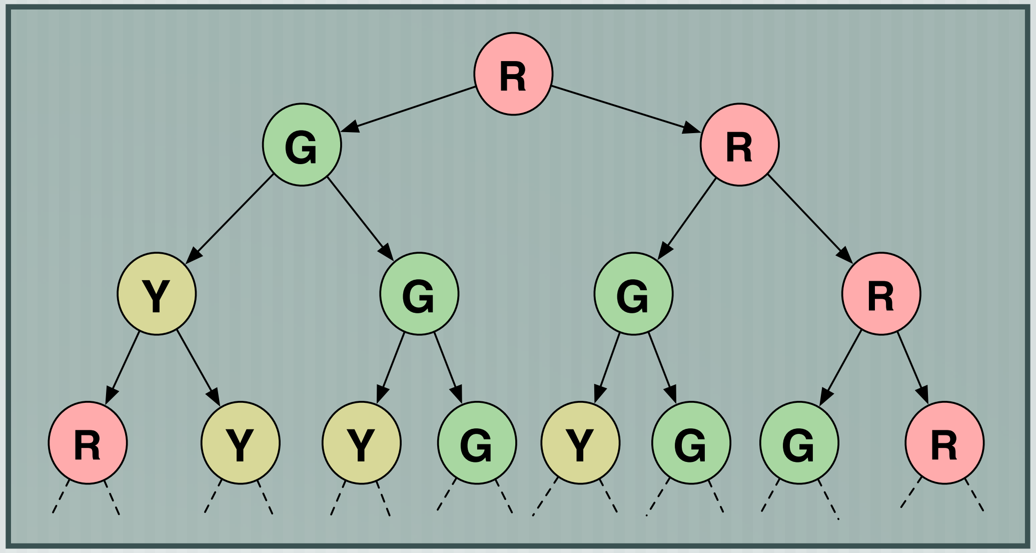
- Operators :
  - X p (next p), G p (always p), F p (p will be true), p U q (p until q), p W q (p weak until q)
  - with quantifiers A (all) or E (eventually)
- Exemple : AG(R ➡ A(R W G))
- Formalism well known and mastered by the model checking community

Expression of the properties : CTL

CTL : Computational Tree Logic

- time is seen as the tree of possible future



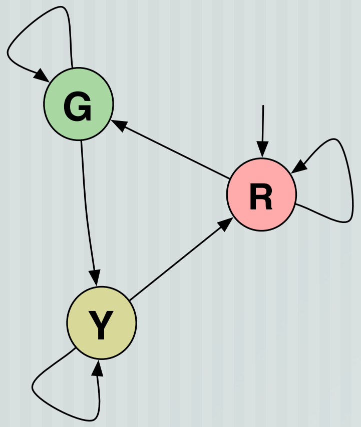


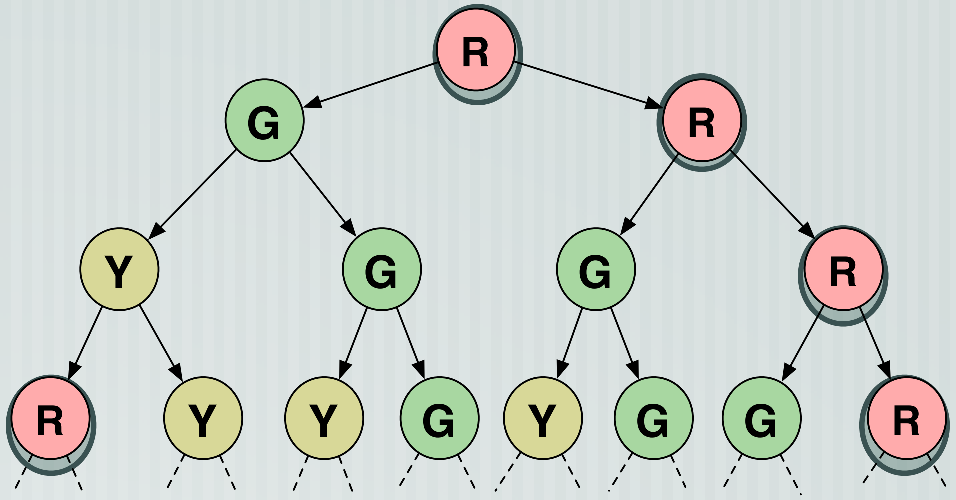
- Operators :
  - X p (next p), G p (always p), F p (p will be true), p U q (p until q), p W q (p weak until q)
  - with quantifiers A (all) or E (eventually)
- Exemple : AG(R ➔ A(R W G))
- Formalism well known and mastered by the model checking community

Expression of the properties : CTL

CTL : Computational Tree Logic

- time is seen as the tree of possible future



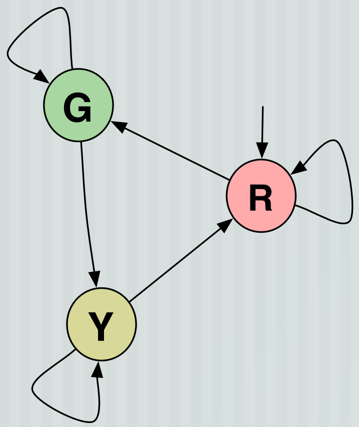


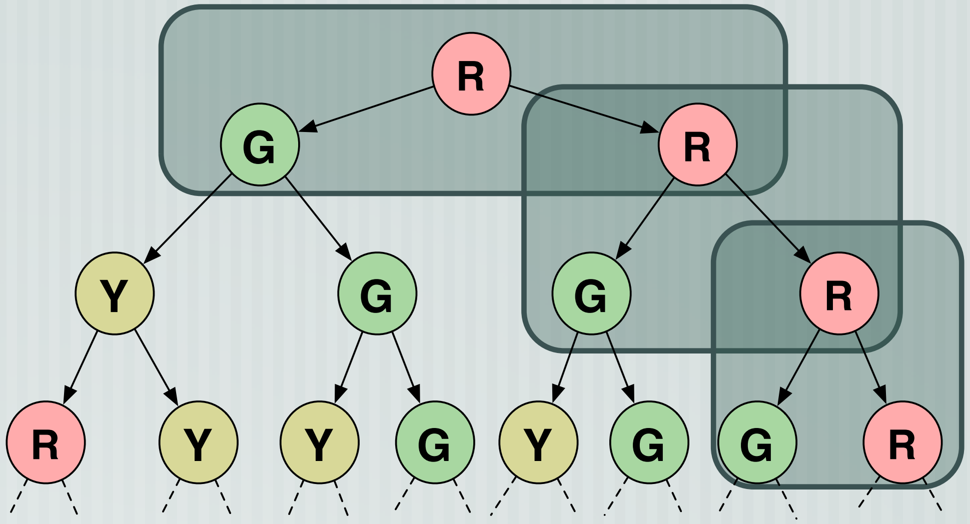
- Operators :
  - X p (next p), G p (always p), F p (p will be true), p U q (p until q), p W q (p weak until q)
  - with quantifiers A (all) or E (eventually)
- Exemple : AG(R ➡ A(R W G))
- Formalism well known and mastered by the model checking community

Expression of the properties : CTL

CTL : Computational Tree Logic

- time is seen as the tree of possible future





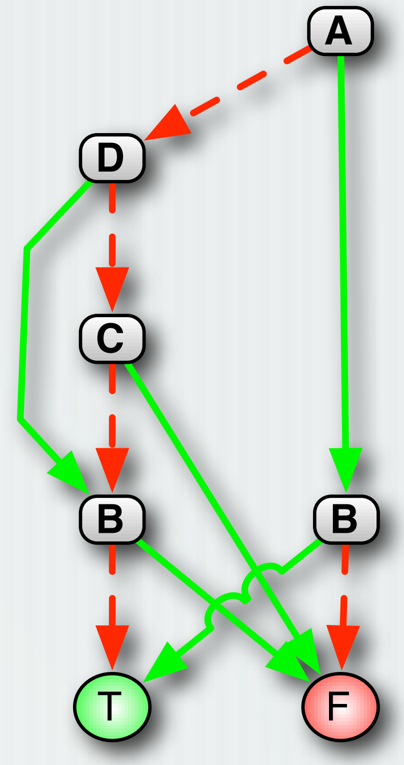
- Operators :
  - X p (next p), G p (always p), F p (p will be true), p U q (p until q), p W q (p weak until q)
  - with quantifiers A (all) or E (eventually)
- Exemple : AG(R ➡ A(R W G))

If the light is red
it will remain red or go green

- Formalism well known and mastered by the model checking community

OBDDs

- OBDD : Ordered Binary Decision Diagram
- Binary graph where nodes correspond to a binary test
- Canonical and compact form
- Drawbacks :
  - sensitive to variables order
  - only symbolic variables

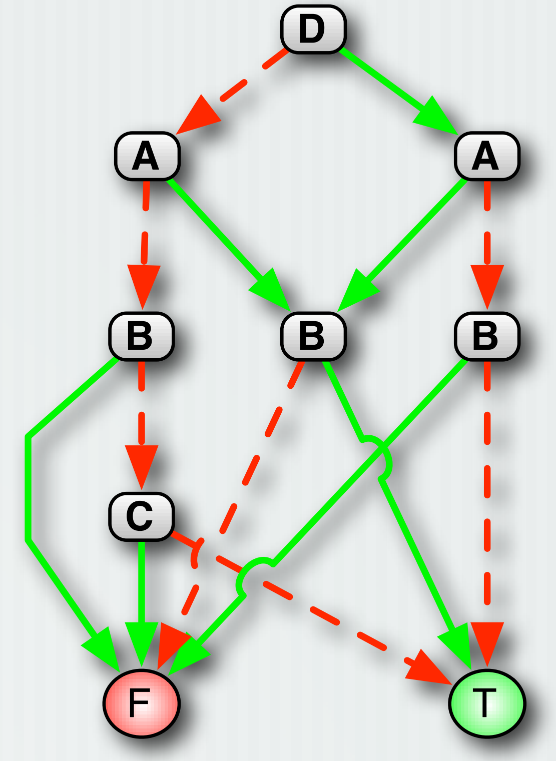


$(a=b)\wedge(b\vee(c\Rightarrow d))$

[Bryant 86] R.E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation. Transactions on Computers, 1986.
[Burch 92] J.R. Burch et al., Symbolic Model Checking : 10^{20} States and Beyond. Information & Computing, 1992.

OBDDs

- OBDD : Ordered Binary Decision Diagram
- Binary graph where nodes correspond to a binary test
- Canonical and compact form
- Drawbacks :
  - sensitive to variables order
  - only symbolic variables

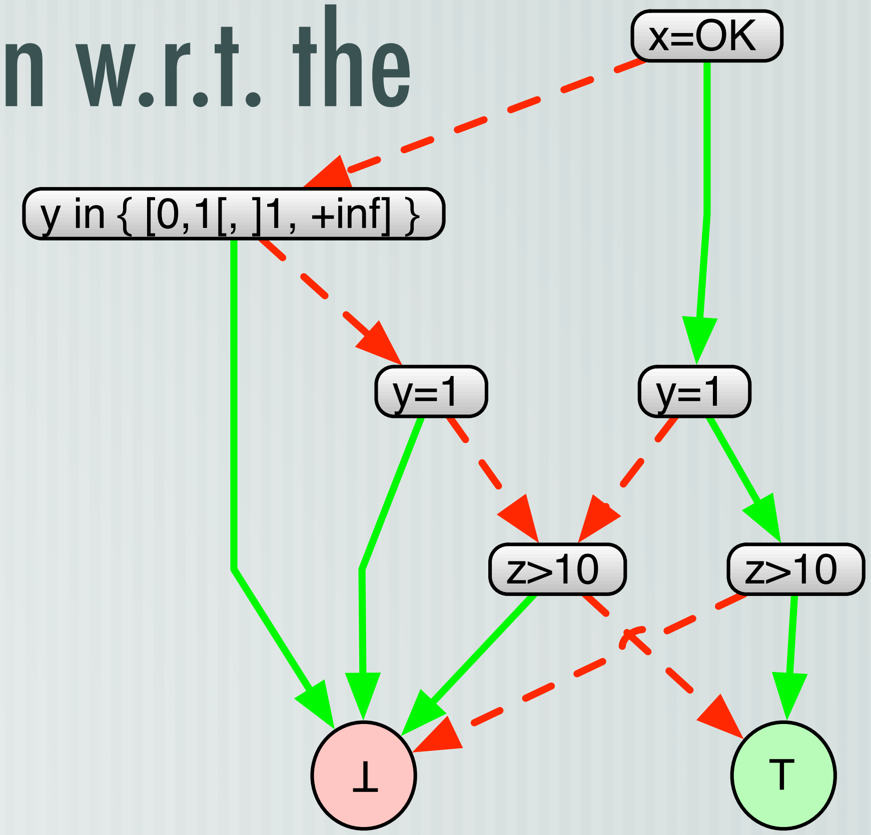


$(a=b)\wedge(b\vee(c\Rightarrow d))$

[Bryant 86] R.E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation. Transactions on Computers, 1986.
[Burch 92] J.R. Burch et al., Symbolic Model Checking : 10^{20} States and Beyond. Information & Computing, 1992.

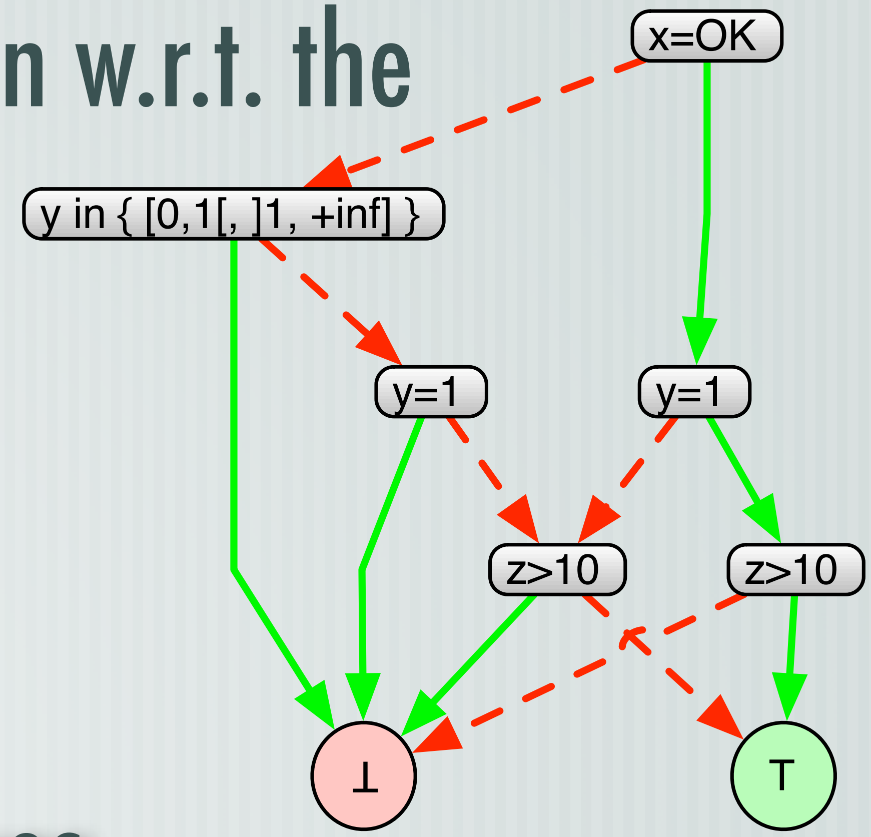
OCRDs

- Similar to OBDDs
- variables have a fixed constraint associated : e.g. y in [0.0, +∞[
- for each variable we create a partition w.r.t. the constraints
- result is equivalent to an OBDD



OCRDs

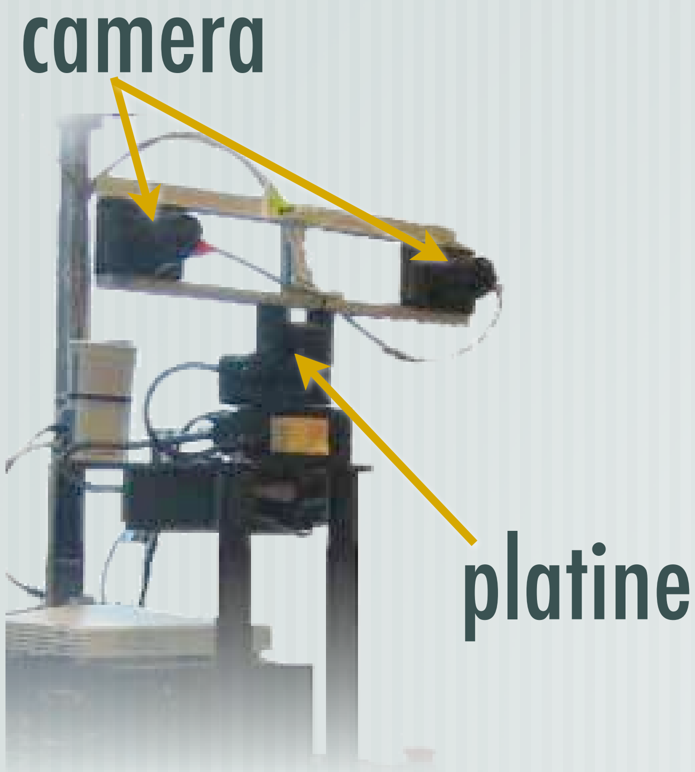
- Similar to OBDDs
- variables have a fixed constraint associated : e.g. y in [0.0, +∞[
- for each variable we create a partition w.r.t. the constraints
- result is equivalent to an OBDD



We can express fixed constraints on request arguments and report values.

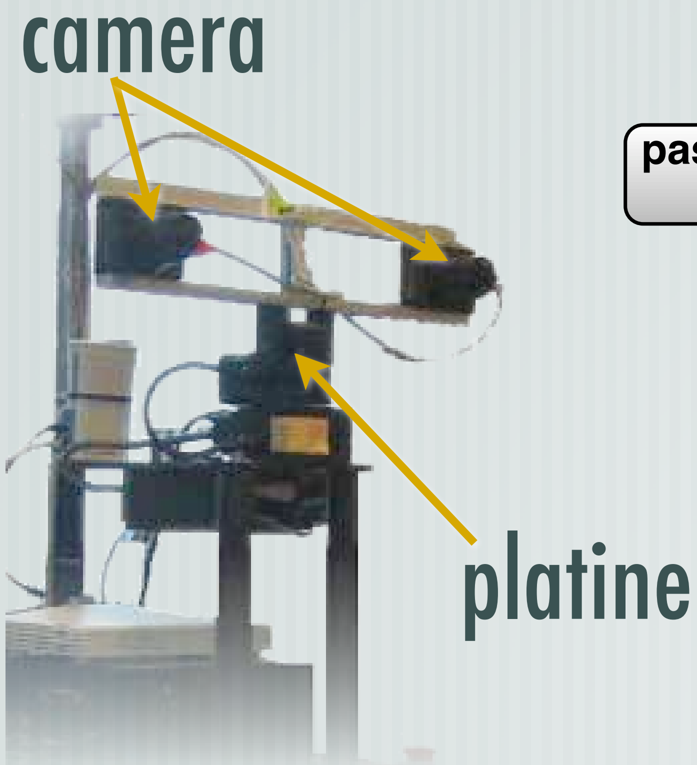
Example

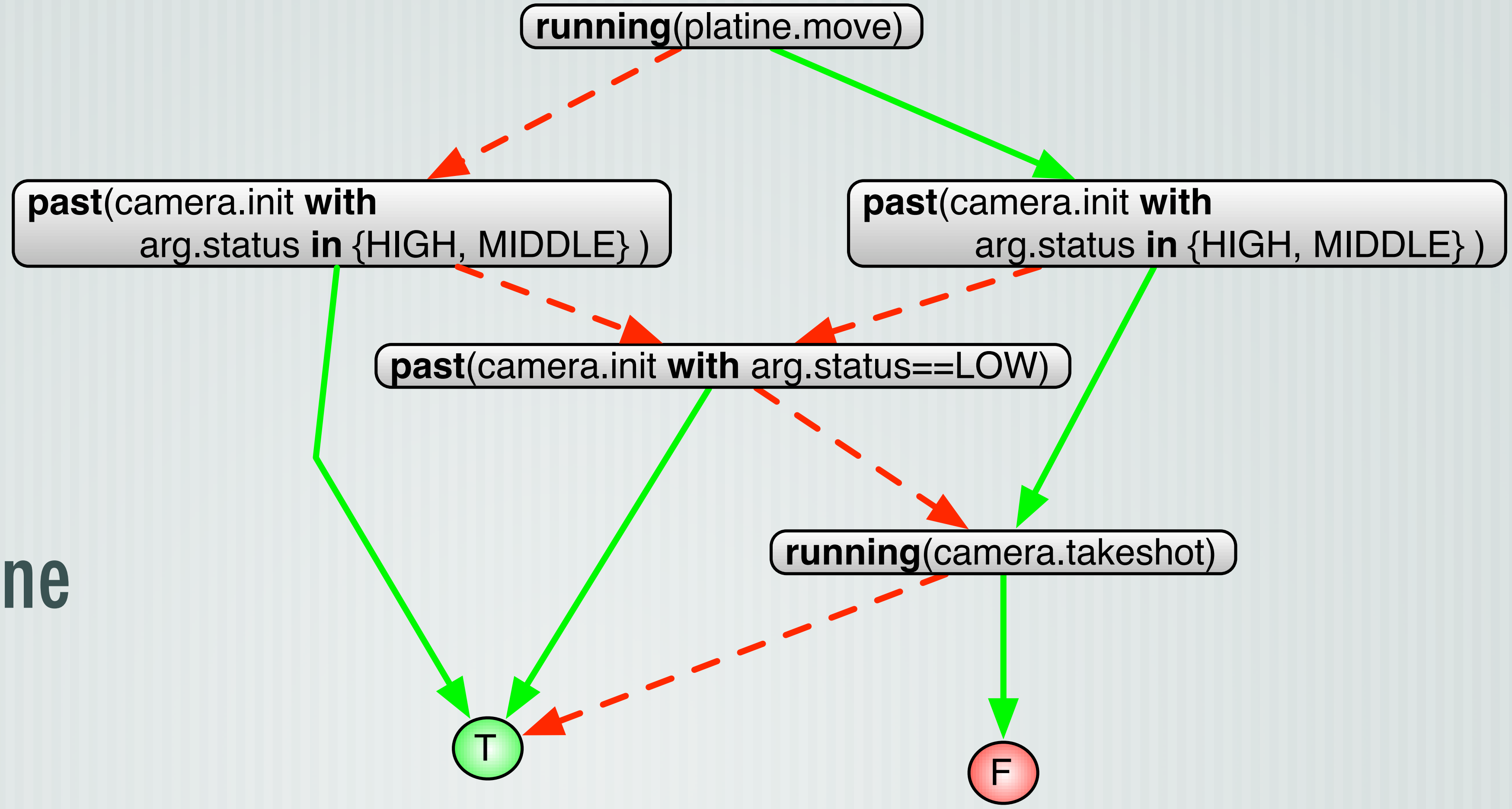
```
check {
never: running(camera.takeshot()) && !last(camera.init(?mode));
always: last(camera.init(?mode) with ?mode!=LOW)
            => !( running(platine.move(?pos))
                    && running(camera.takeshot()) );
}
```



Example

```
check {
never: running(camera.takeshot()) && !last(camera.init(?mode));
always: last(camera.init(?mode) with ?mode!=LOW)
          => !( running(platine.move(?pos))
                && running(camera.takeshot()) );
}
```





running(platine.move)

past(camera.init **with** arg.status **in** {HIGH, MIDDLE})

past(camera.init **with** arg.status **in** {HIGH, MIDDLE})

past(camera.init **with** arg.status==LOW)

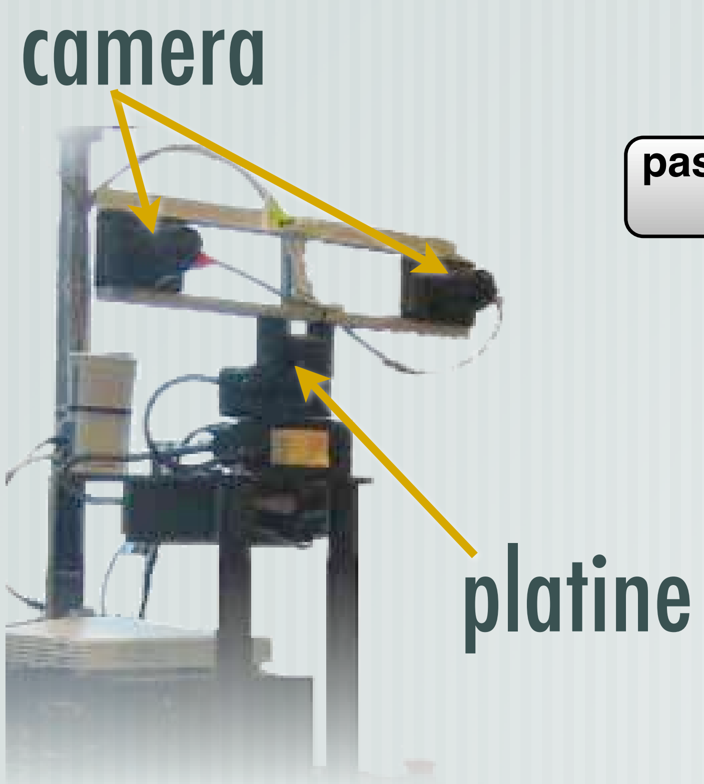
running(camera.takeshot)

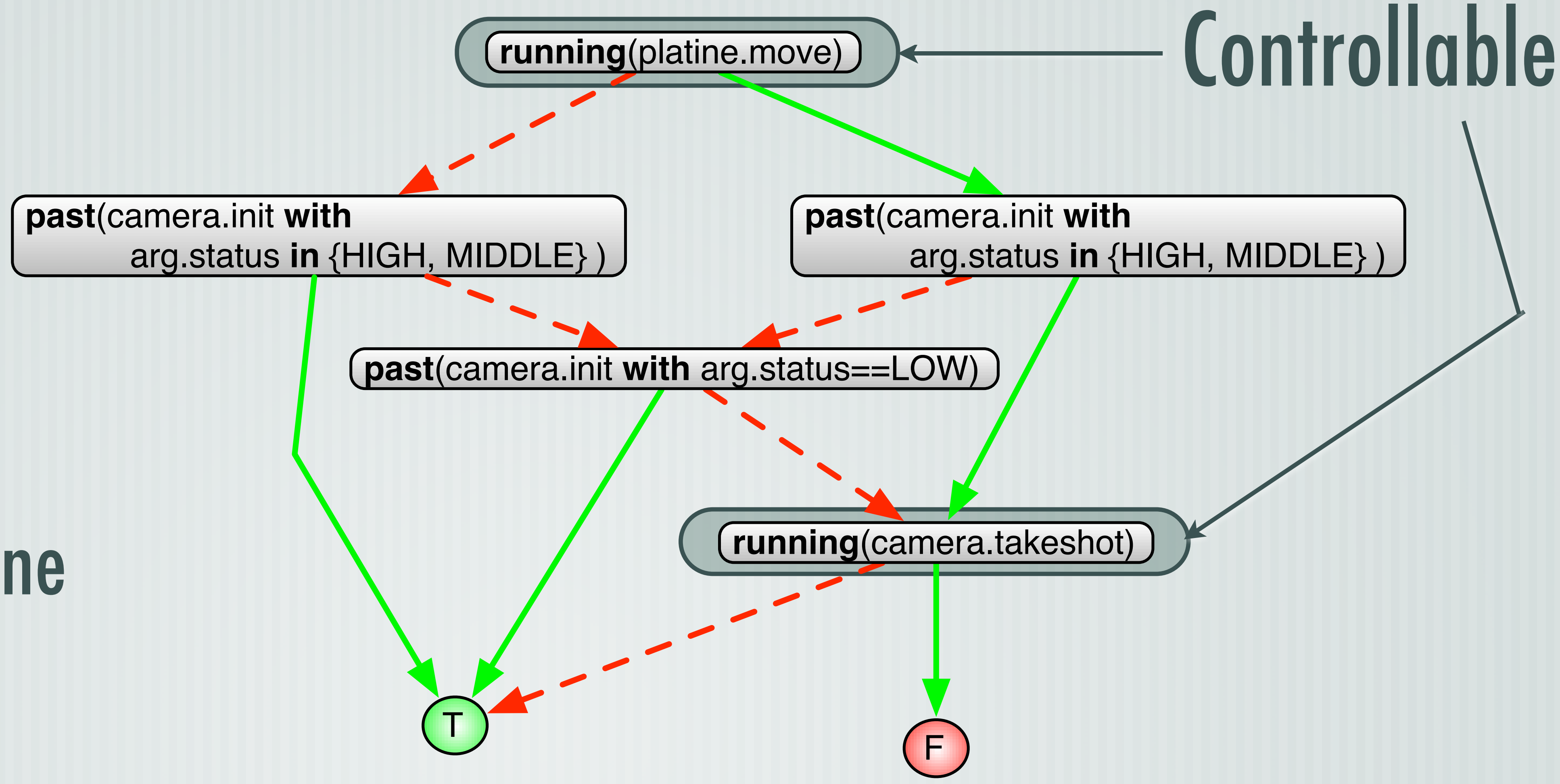
T

F

Example

```
check {
never: running(camera.takeshot()) && !last(camera.init(?mode));
always: last(camera.init(?mode) with ?mode!=LOW)
          => !( running(platine.move(?pos))
                && running(camera.takeshot()) );
}
```




running(platine.move)

Controllable

past(camera.init with arg.status in {HIGH, MIDDLE})

past(camera.init with arg.status in {HIGH, MIDDLE})

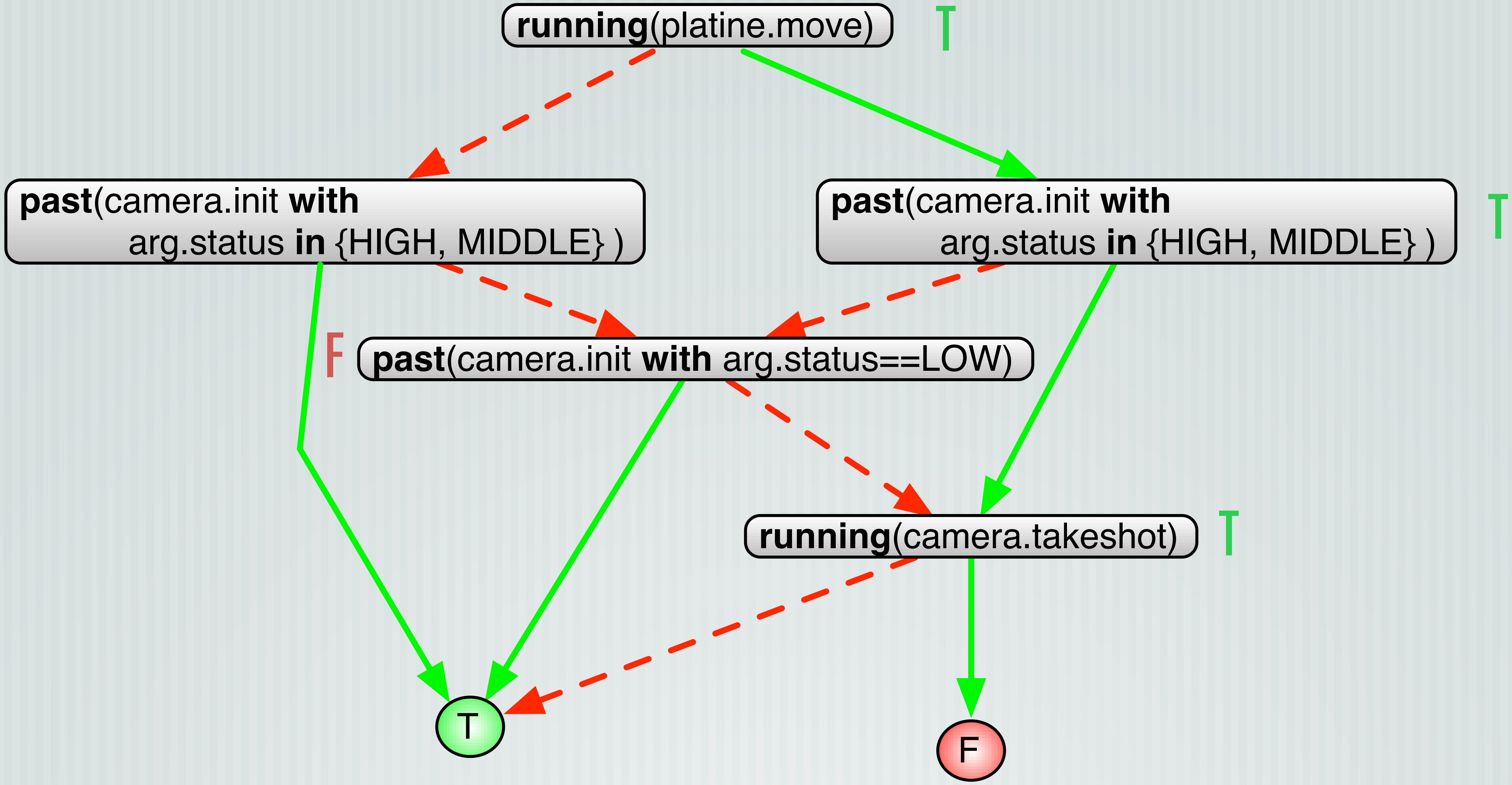
past(camera.init with arg.status==LOW)

running(camera.takeshot)

T

F

State Checker



running(platine.move)

T

past(camera.init **with** arg.status **in** {HIGH, MIDDLE})

past(camera.init **with** arg.status **in** {HIGH, MIDDLE})

T

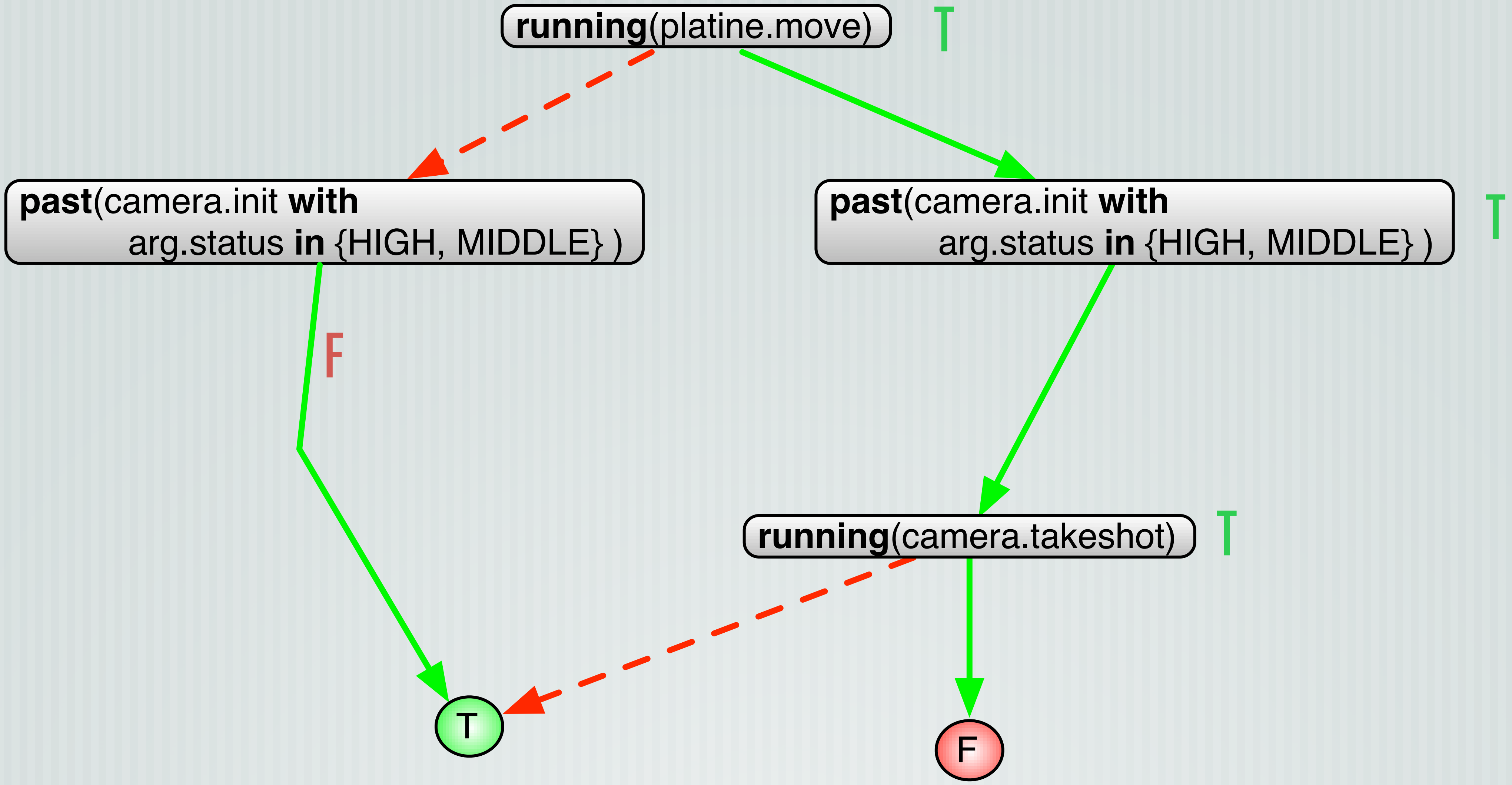
F

past(camera.init **with** arg.status==LOW)

running(camera.takeshot)

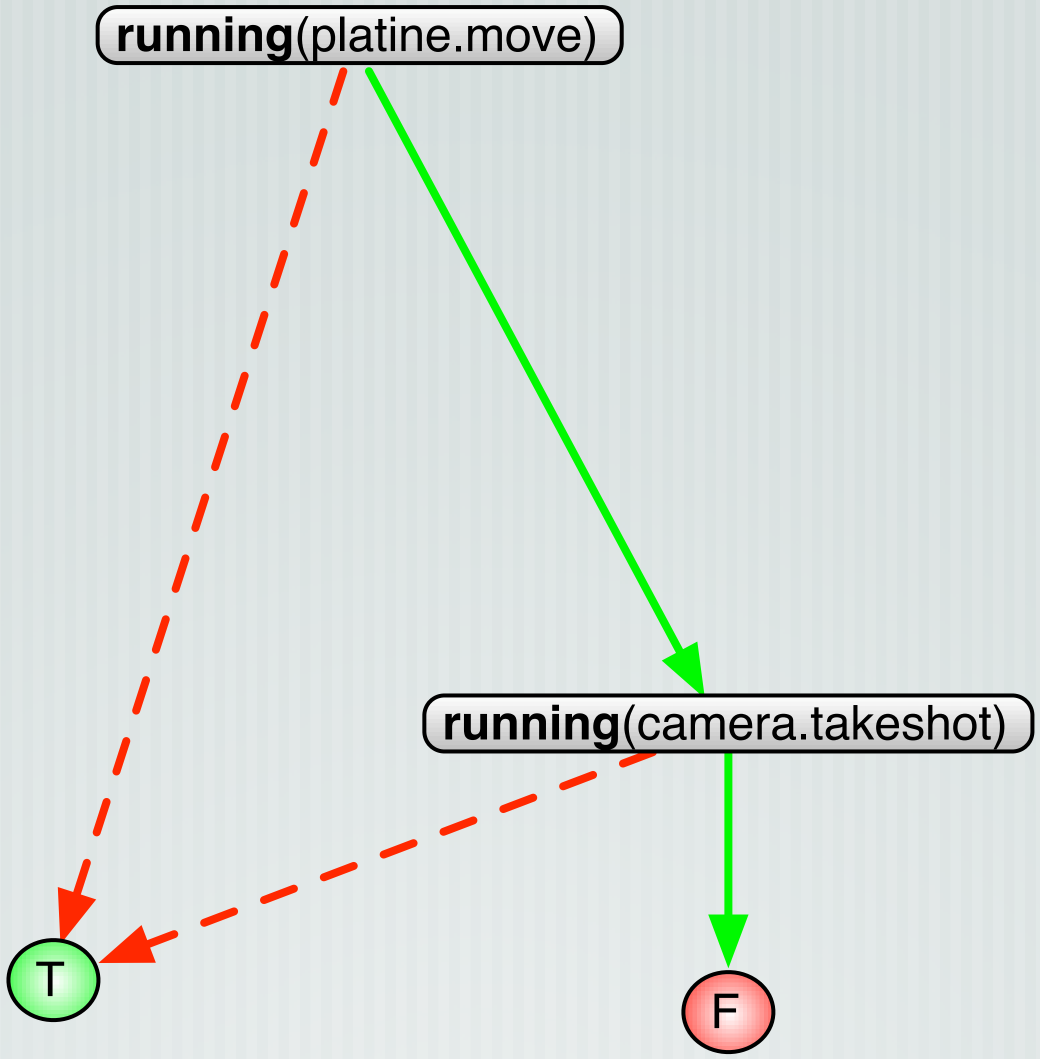
T

State Checker


running(platine.move)
T
past(camera.init **with** arg.status **in** {HIGH, MIDDLE})
past(camera.init **with** arg.status **in** {HIGH, MIDDLE})
T
F
running(camera.takeshot)
T
T
F

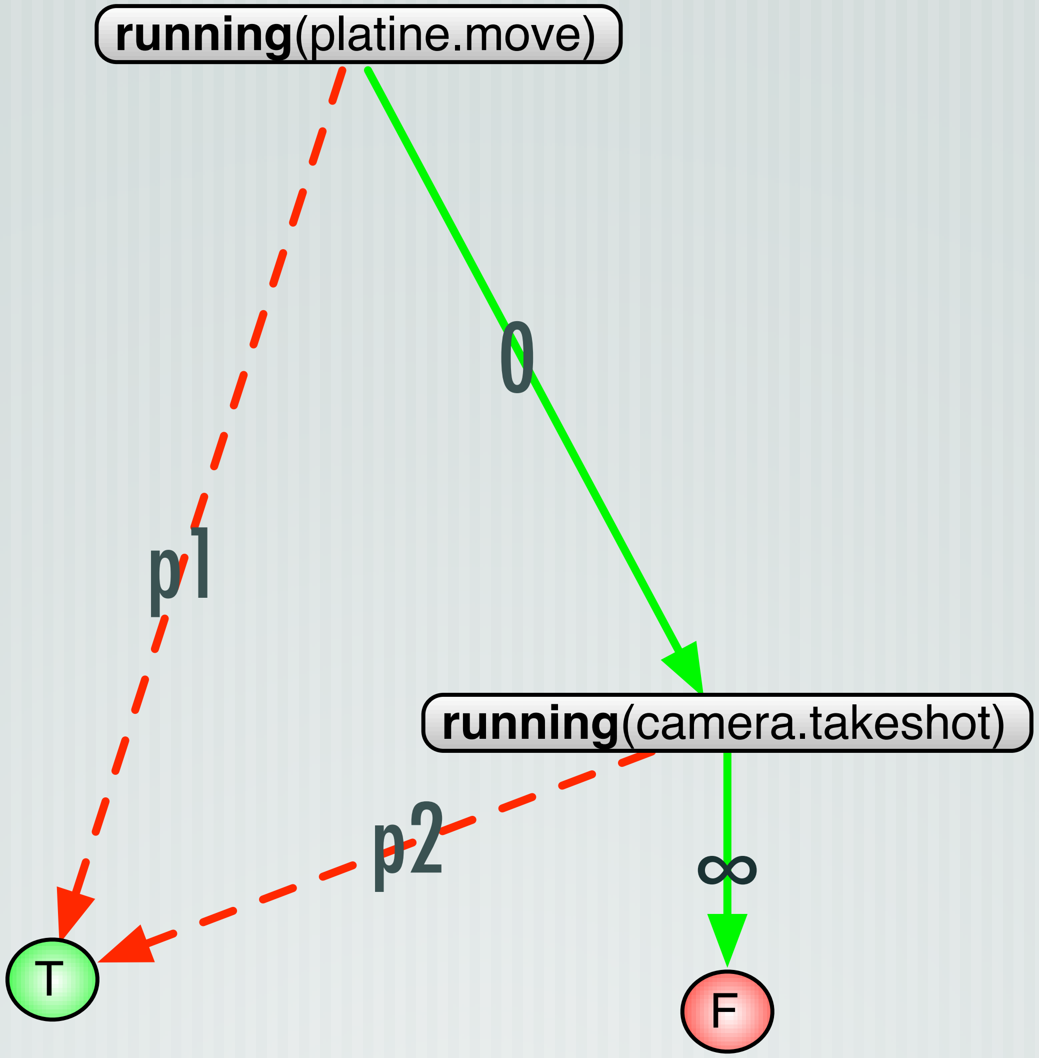
1- We set the non controllable predicates

State Checker



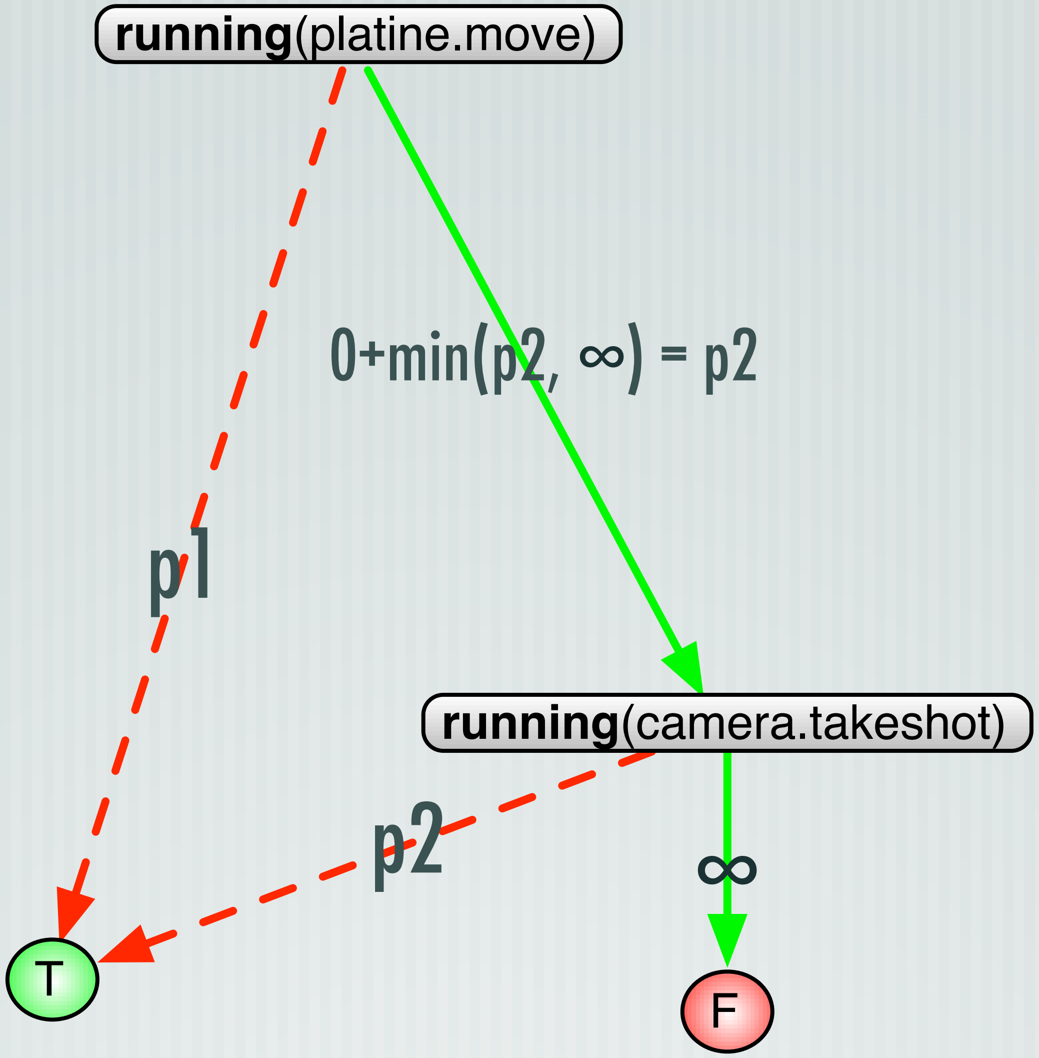
1- We set the non controllable predicates

State Checker



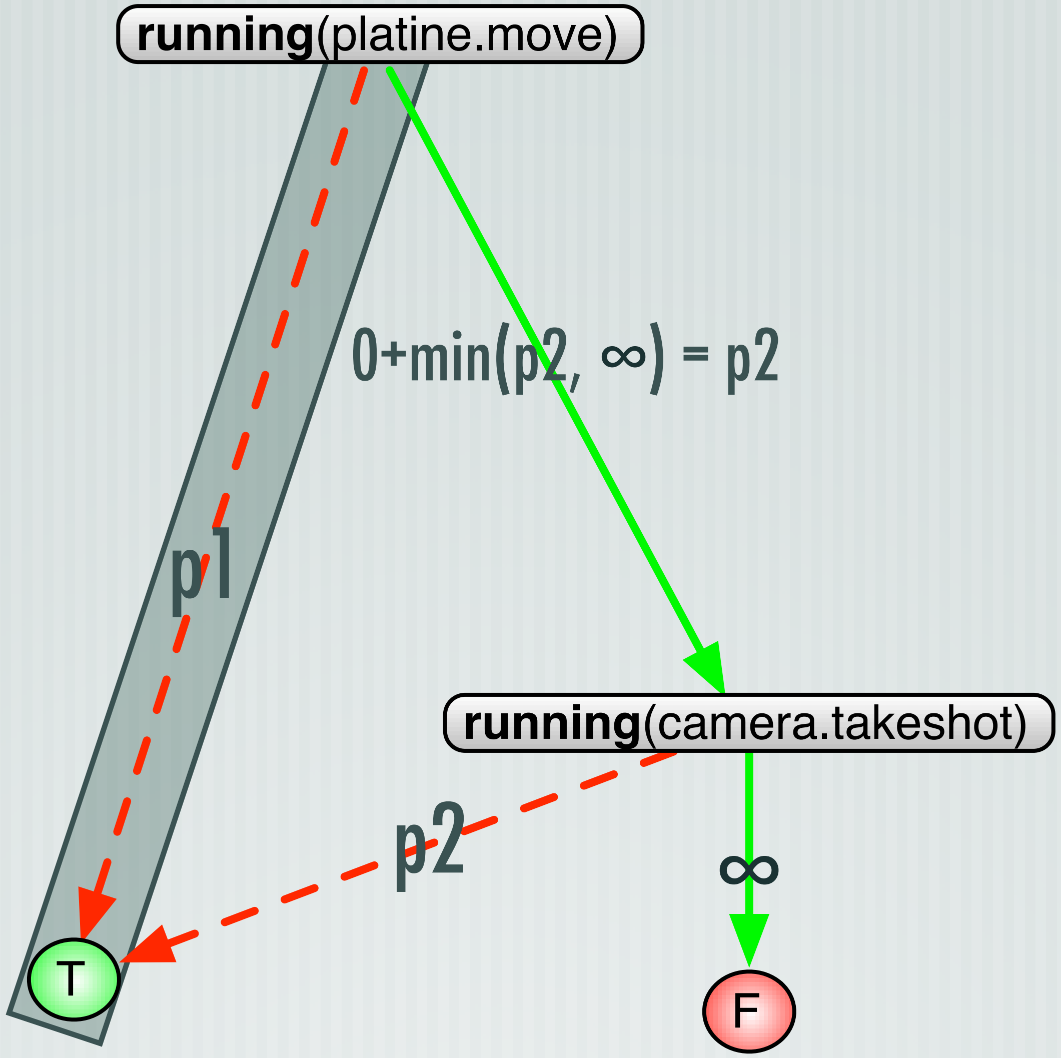
2- We evaluate the cost of various solution

State Checker



2- We evaluate the cost of various solution

State Checker

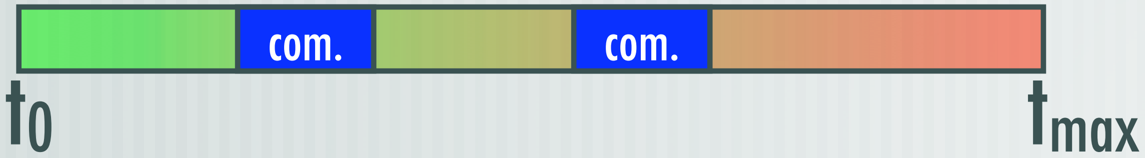


3- We choose the less expensive solution (p1<p2)

Test With An Autonomous Robot

Plan (IxTeT-Exec) :

- Objective : take science pictures in a time frame
- Repair or replan when problems occur
- Adding goal during communication window



Constraints on Dala

```
#define refME "rflex"

check {
 always: ( running(pom.addME) || running(pom.addSE) )  => last(pom.SetModel);
 never: running(pom.SetRefME with arg.name!=refME);
 always: running(pom.setRefME) => last(pom.addME with arg.name==refME);
 always: running(pom.Run) => last(pom.setRefME);
 always: running(ndd.GoTo) => last(pom.Run);
 always: running(ndd.GoTo) => ( last(ndd.SetParams) && last(ndd.SetSpeed with arg.linear<1.0 ) );
 always: running(ndd.GoTo) => running(aspect.AspectFromPosterConfig);
 always: running(aspect.AspectFromPosterConfig) => last(aspect.SetViewParameters);
 always: running(aspect.AspectFromPosterConfig with
                arg.posPosterName.name.name=="pomSickFramePos") => last(sick.SetPomTagging with arg==SICK_TRUE);
always: running(aspect.AspectFromPosterConfig with
                arg.posPosterName.name.name=="pomSickFramePos") => last(sick.ContinuousShot);
 never: running(antenna.Comunicate) && running(rflex.TrackSpeedStart);
 never: running(rflex.TrackSpeedStart) && ( running(platine.CmdPosCoord) || running(platine.CmdPosPan)
                                        || running(platine.CmdPosTilt) || running(platine.TrackPos) );
 never: running(rflex.TrackSpeedStart with arg.name.value.v>0.9);
 always: running(antenna.Comunicate) => last(antenna.AddWindow);
 always: running(antenna.AddWindow) => last(antenna.Init);
 always: running(camera.OneShot) => last(camera.Initialize);
 never: running(camera.OneShot) &&( running(platine.CmdPosCoord) || running(platine.CmdPosPan)
                                   || running(platine.CmdPosTilt)|| running(platine.TrackPos) );
}
```

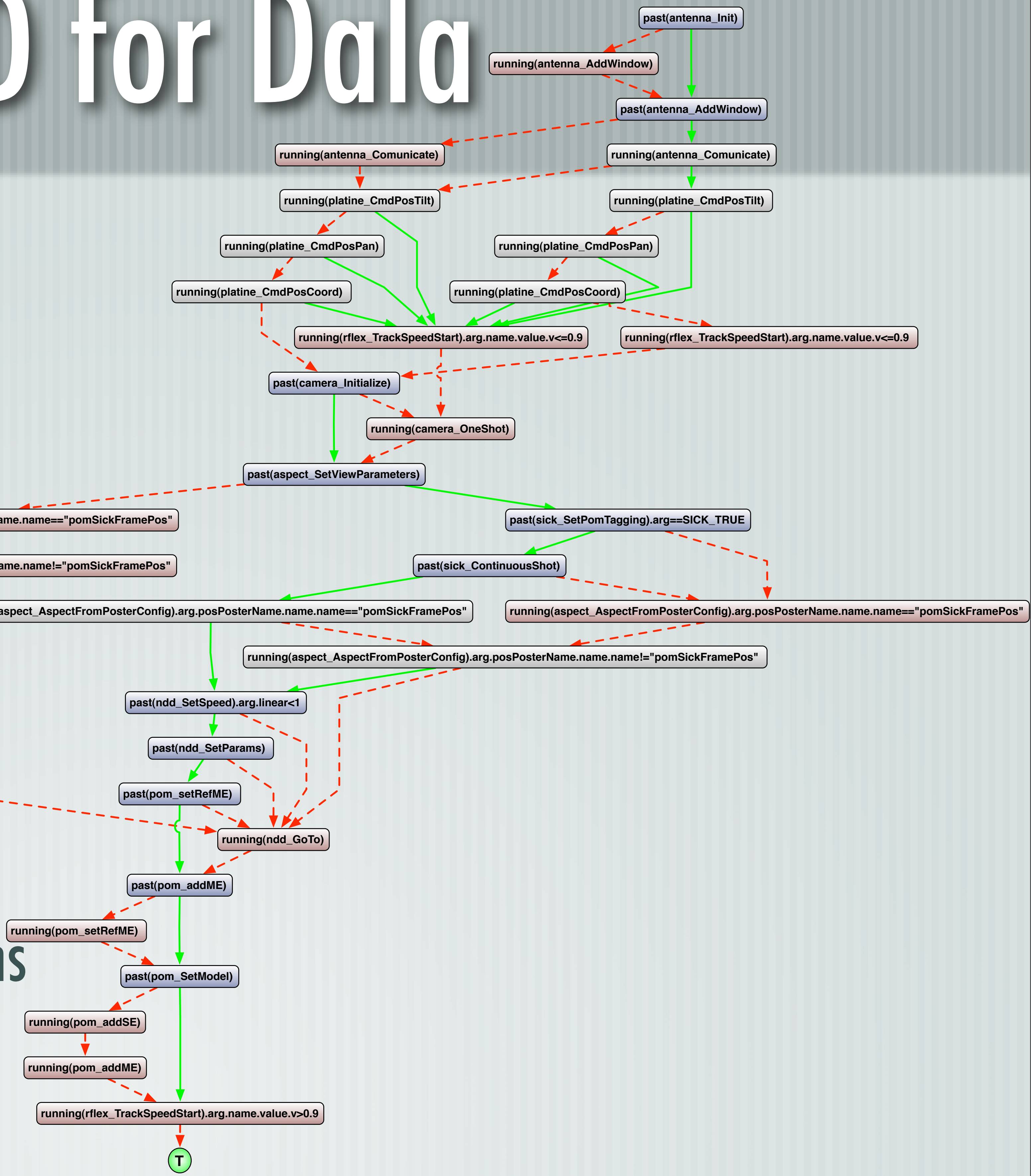
Constraints on Dala

```
#define refME "rflex"

check {
 always: ( running(pom.addME) || running(pom.addSE) )  => last(pom.SetModel);
 never: running(pom.SetRefME with arg.name!=refME);
 always: running(pom.setRefME) => last(pom.addME with arg.name==refME);
 always: running(pom.Run) => last(pom.setRefME);
 always: running(ndd.GoTo) => last(pom.Run);
 always: running(ndd.GoTo) => ( last(ndd.SetParams) && last(ndd.SetSpeed with arg.linear<1.0 ) );
 always: running(ndd.GoTo) => running(aspect.AspectFromPosterConfig);
 always: running(aspect.AspectFromPosterConfig) => last(aspect.SetViewParameters);
 always: running(aspect.AspectFromPosterConfig with
                     arg.posPosterName.name.name=="pomSickFramePos") => last(sick.SetPomTagging with arg==SICK_TRUE);
always: running(aspect.AspectFromPosterConfig with
                     arg.posPosterName.name.name=="pomSickFramePos") => last(sick.ContinuousShot);
 never: running(antenna.Comunicate) && running(rflex.TrackSpeedStart);
 never: running(rflex.TrackSpeedStart) && ( running(platine.CmdPosCoord) || running(platine.CmdPosPan)
                                            || running(platine.CmdPosTilt) || running(platine.TrackPos) );
 never: running(rflex.TrackSpeedStart with arg.name.value.v>0.9);
 always: running(antenna.Comunicate) => last(antenna.AddWindow);
 always: running(antenna.AddWindow) => last(antenna.Init);
 always: running(camera.OneShot) => last(camera.Initialize);
 never: running(camera.OneShot) &&( running(platine.CmdPosCoord) || running(platine.CmdPosPan)
                                      || running(platine.CmdPosTilt) || running(platine.TrackPos) );
}
```

17 rules

1 to 5 predicates per rules

Resulting OCRD for Dala



past(antenna_Init)

running(antenna_AddWindow)

past(antenna_AddWindow)

running(antenna_Comunicate)

running(antenna_Comunicate)

running(platine_CmdPosTilt)

running(platine_CmdPosTilt)

running(platine_CmdPosPan)

running(platine_CmdPosPan)

running(platine_CmdPosCoord)

running(platine_CmdPosCoord)

running(rflex_TrackSpeedStart).arg.name.value.v<=0.9

running(rflex_TrackSpeedStart).arg.name.value.v<=0.9

past(camera_Initialize)

running(camera_OneShot)

past(aspect_SetViewParameters)

running(aspect_AspectFromPosterConfig).arg.posPosterName.name.name=="pomSickFramePos"

past(sick_SetPomTagging).arg==SICK_TRUE

running(aspect_AspectFromPosterConfig).arg.posPosterName.name.name!="pomSickFramePos"

past(sick_ContinuousShot)

running(aspect_AspectFromPosterConfig).arg.posPosterName.name.name=="pomSickFramePos"

running(aspect_AspectFromPosterConfig).arg.posPosterName.name.name=="pomSickFramePos"

running(aspect_AspectFromPosterConfig).arg.posPosterName.name.name!="pomSickFramePos"

past(ndd_SetSpeed).arg.linear<1

past(ndd_SetParams)

past(pom_setRefME)

running(ndd_GoTo)

past(pom_addME)

running(pom_setRefME)

past(pom_SetModel)

running(pom_addSE)

running(pom_addME)

running(rflex_TrackSpeedStart).arg.name.value.v>0.9

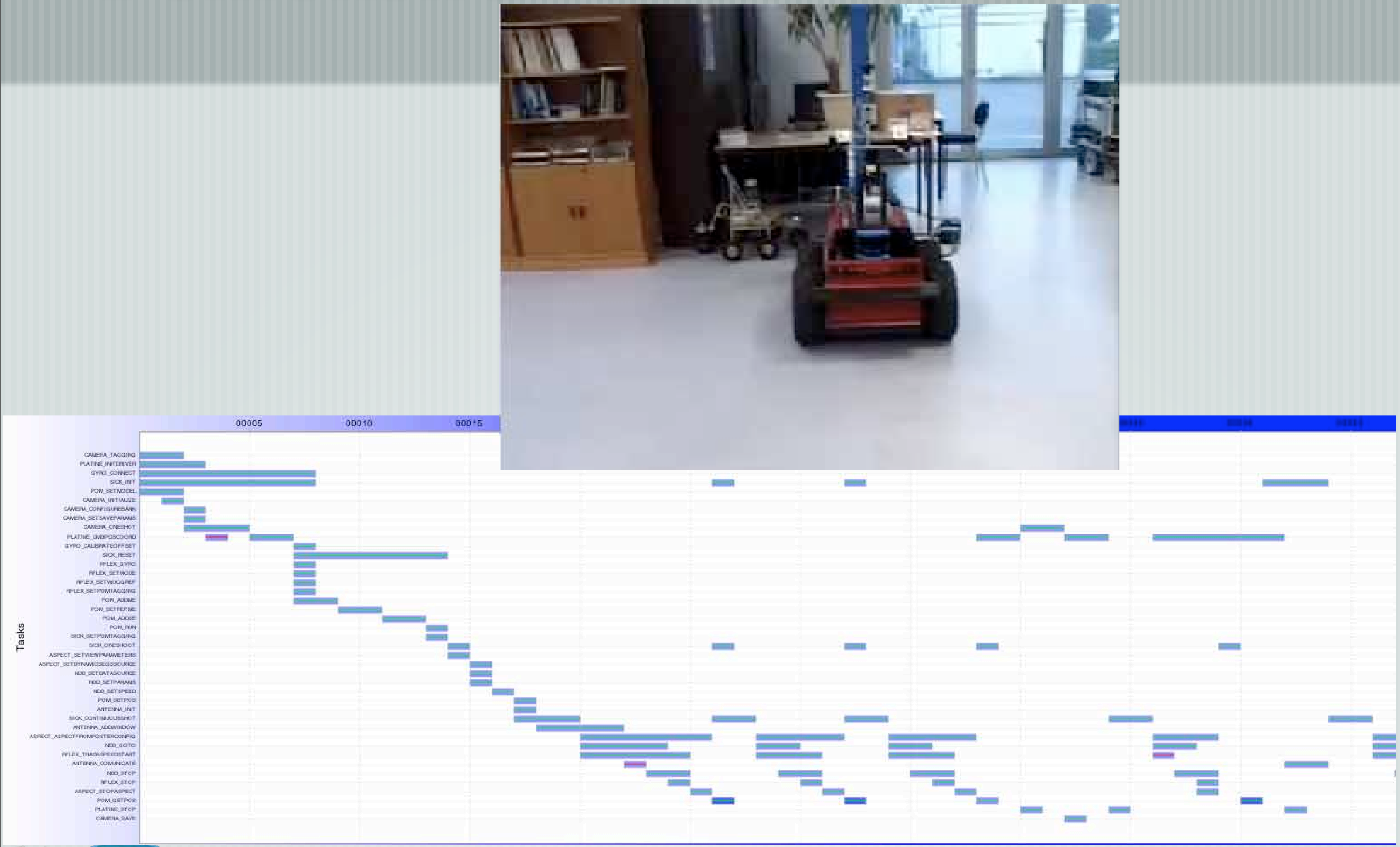
T

Depth : 28

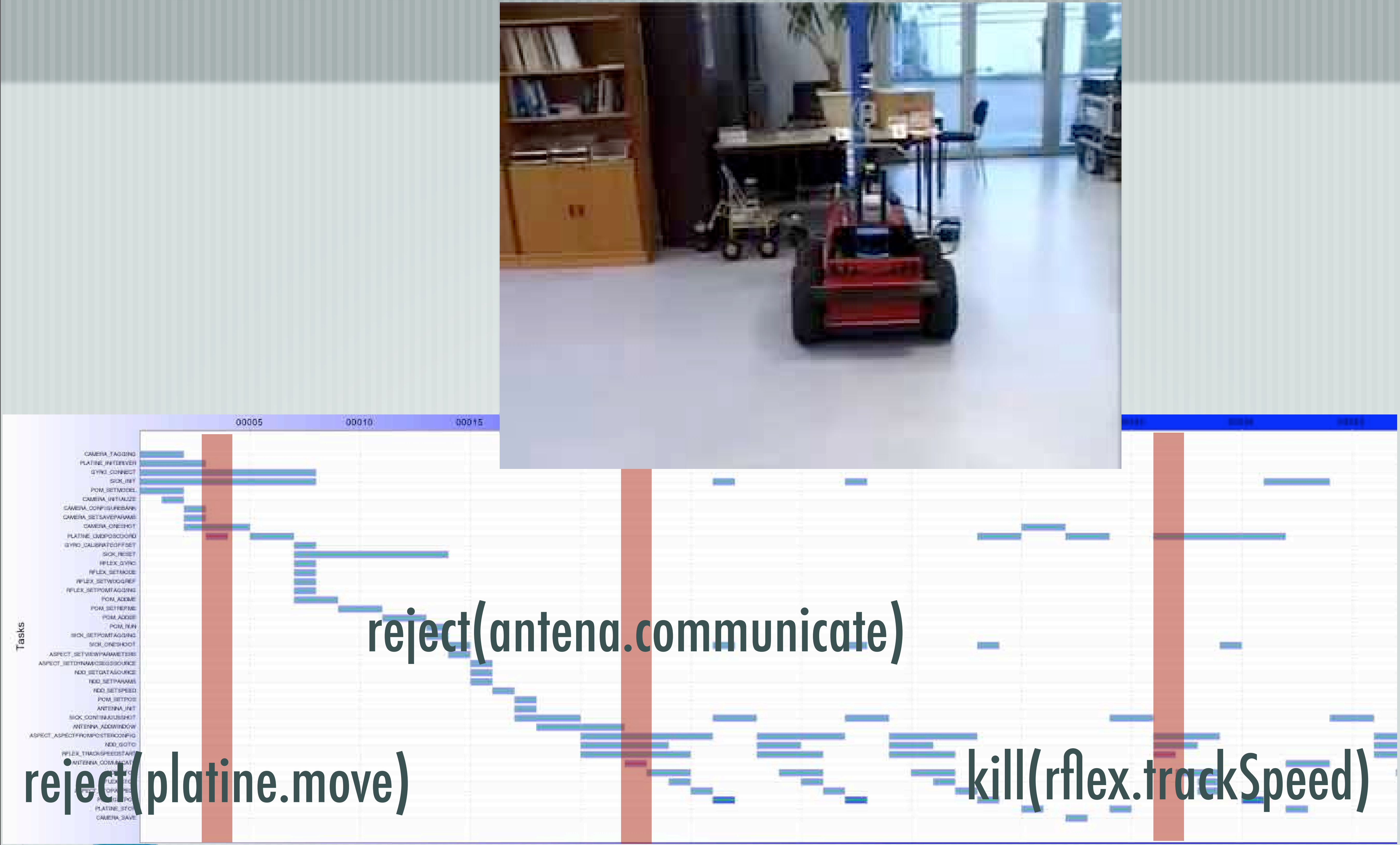
# of nodes : 33

Processing time of the R^2C < 0.4ms

Demonstration ...



Demonstration ...



reject(antena.communicate)

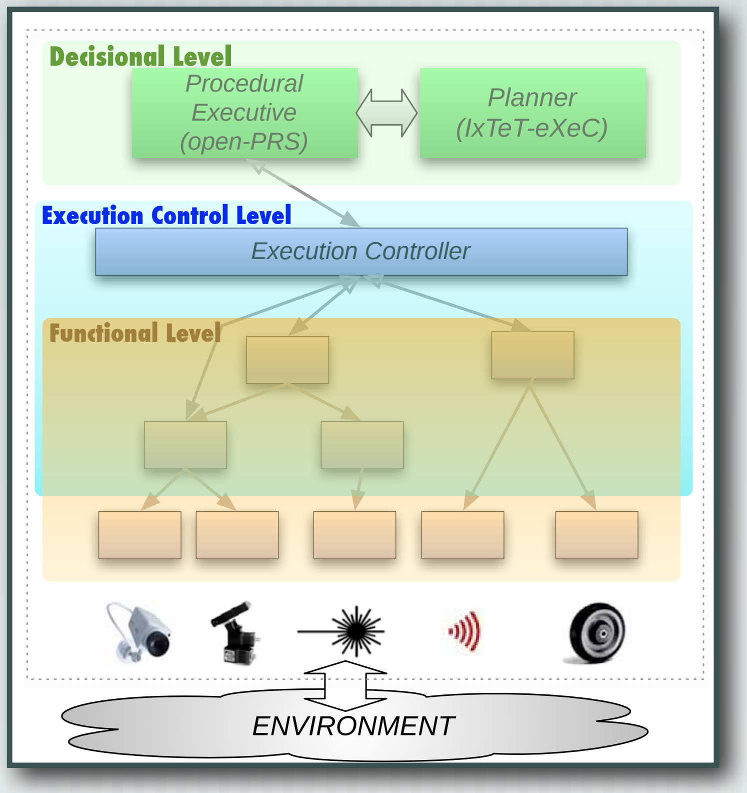
reject(platine.move)

kill(rflex.trackSpeed)

R^{2}C on Dala: Usage Analysis

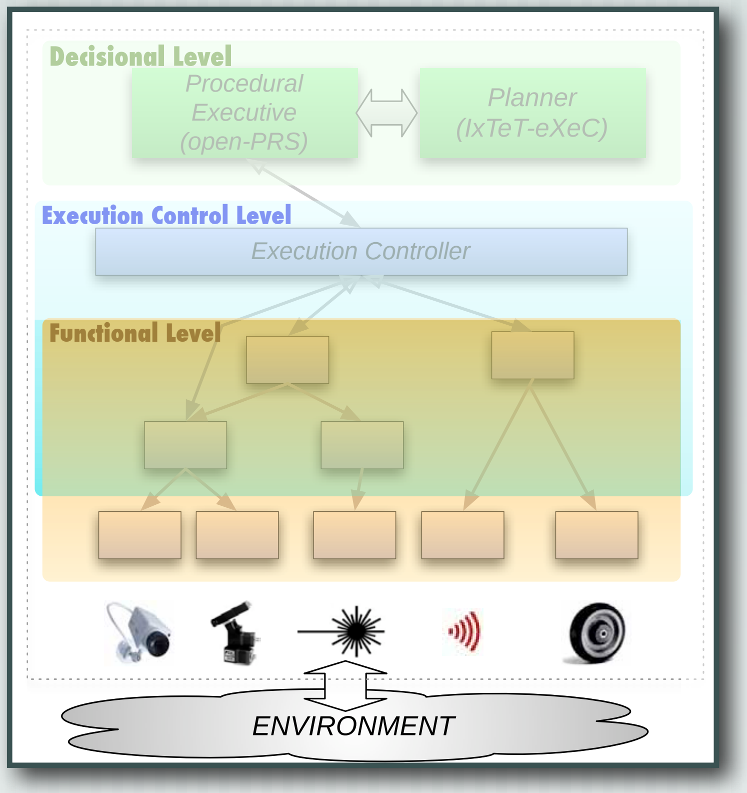
- Detected “real” problems in the Procedural Executive procedures
- Online control => enforces some dependability at all time
- No noticeable effect on the performance of the system

Conclusion


Decisional Level
Procedural Executive (open-PRS)
Planner (IxTeT-eXeC)
Execution Control Level
Execution Controller
Functional Level
ENVIRONMENT

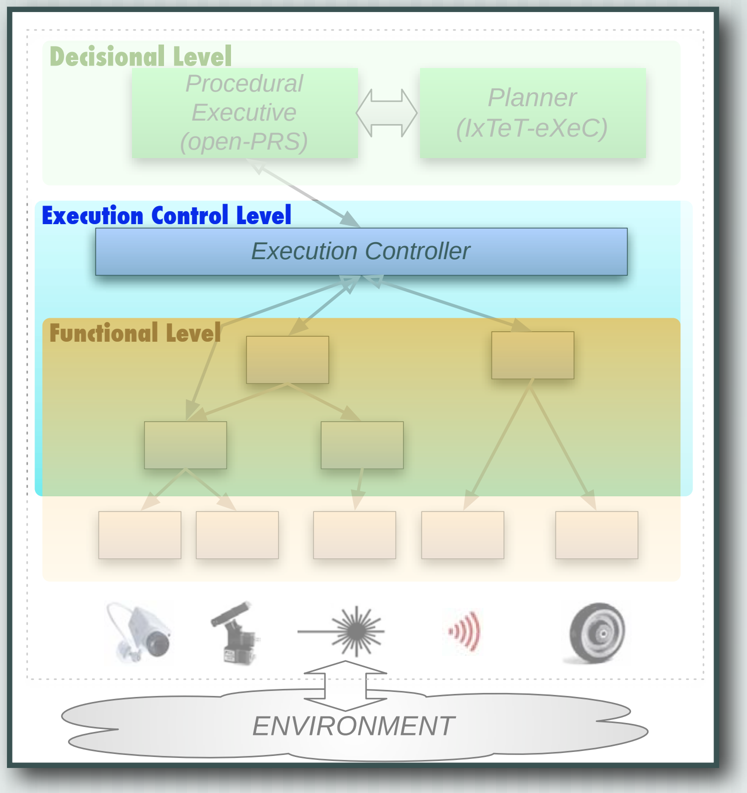
Conclusion

- Functional
  - semi formal framework
  - reusability
  - ease of integration
  - Tool: GenoM



Conclusion

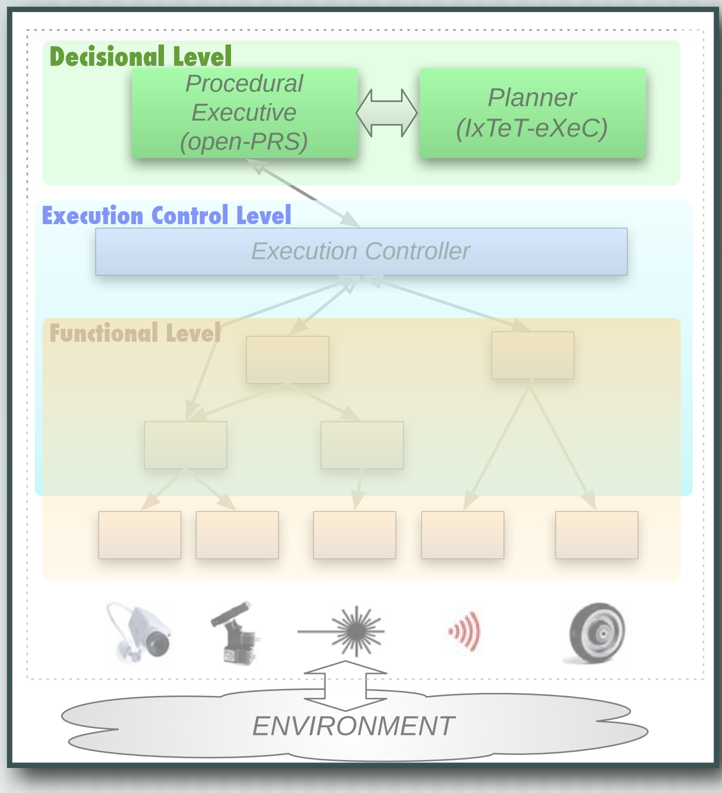
- Functional
  - semi formal framework
  - reusability
  - ease of integration
  - Tool: GenoM



- Execution Control
  - Fault tolerance (safety bag)
  - Tools: R2C

Conclusion

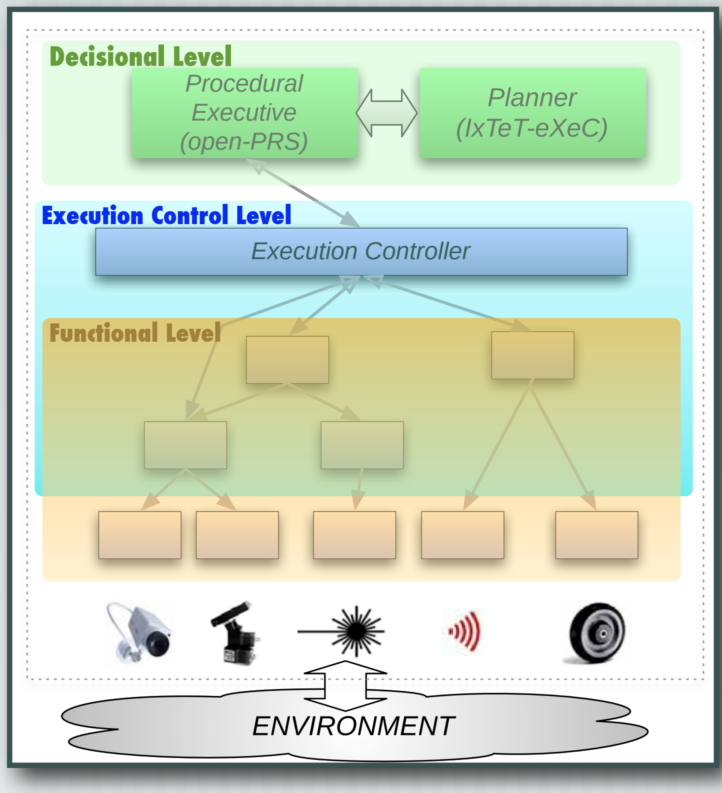
- Functional
  - semi formal framework
  - reusability
  - ease of integration
  - Tool: GenoM
- Execution Control
  - Fault tolerance (safety bag)
  - Tools: R2C



- Decisional
  - bring the autonomy
  - procedural executive
  - planning possible...
  - ... plan execution control is then desirable
  - Tools: OpenPRS, IxTeT

Conclusion

- Functional
  - semi formal framework
  - reusability
  - ease of integration
  - Tool: GenoM
- Execution Control
  - Fault tolerance (safety bag)
  - Tools: R2C



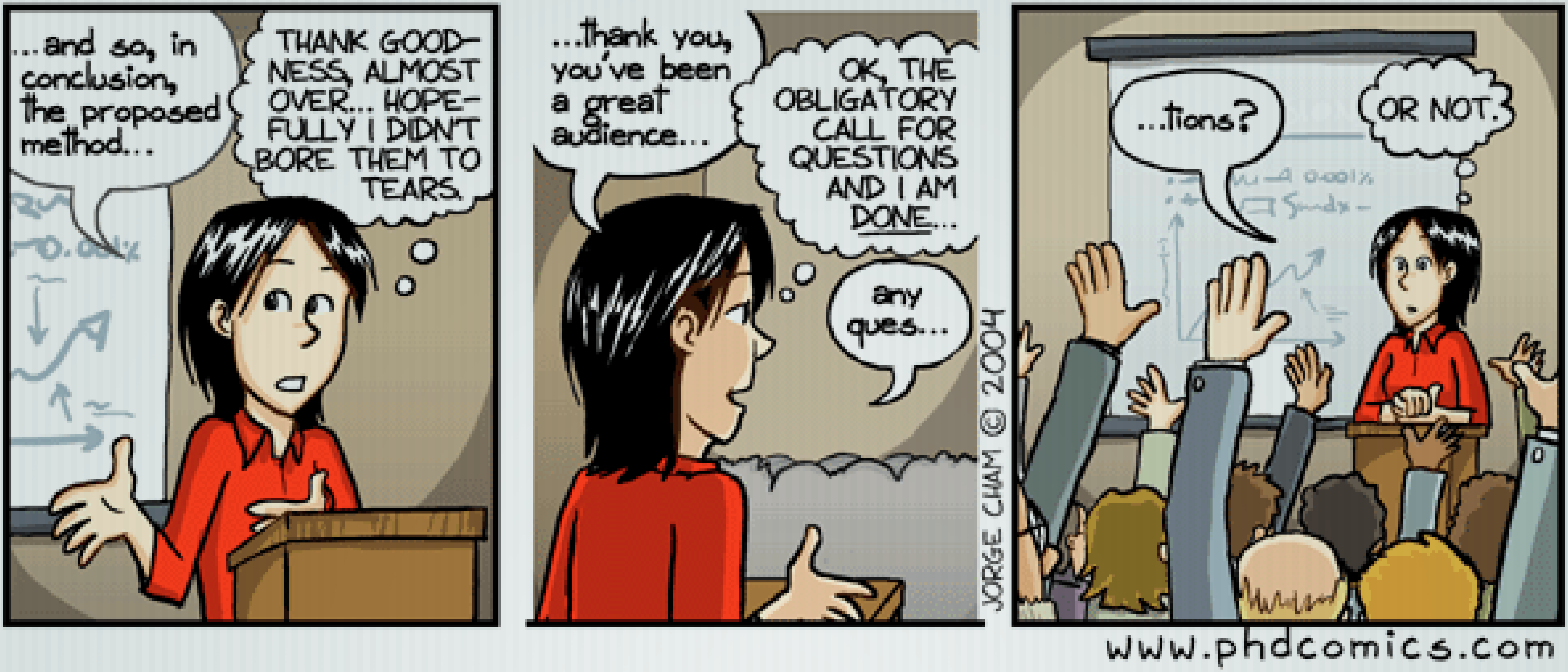
- Decisional
  - bring the autonomy
  - procedural executive
  - planning possible...
  - ... plan execution control is then desirable
  - Tools: OpenPRS, IxTeT

http://softs.laas.fr/

Related Ongoing Works (LAAS)

- SAC (Critical Autonomous System) project (presented by David Powell)
- Safety Bag “rules” specifications (PhD LAAS)
- AMAES (Advanced Methods for Autonomous Embedded Systems, LAAS, Verimag), timed automata (functional and decisional level)
- AGATA (architecture for autonomous satellite)
- COGNIRON (Cognitive Robot EEC FP6 Project)

Thanks


...and so, in conclusion, the proposed method...
THANK GOODNESS, ALMOST OVER... HOPEFULLY I DIDN'T BORE THEM TO TEARS.
...thank you, you've been a great audience...
OK, THE OBLIGATORY CALL FOR QUESTIONS AND I AM DONE...
any ques...
JORGE CHAM © 2004
...tions?
OR NOT.
www.phdcomics.com