Architecture for Autonomy: from Robots to Satellites

Sara Fleury & Félix Ingrand

(sara@laas.fr felix@laas.fr)

LAAS-CNRS

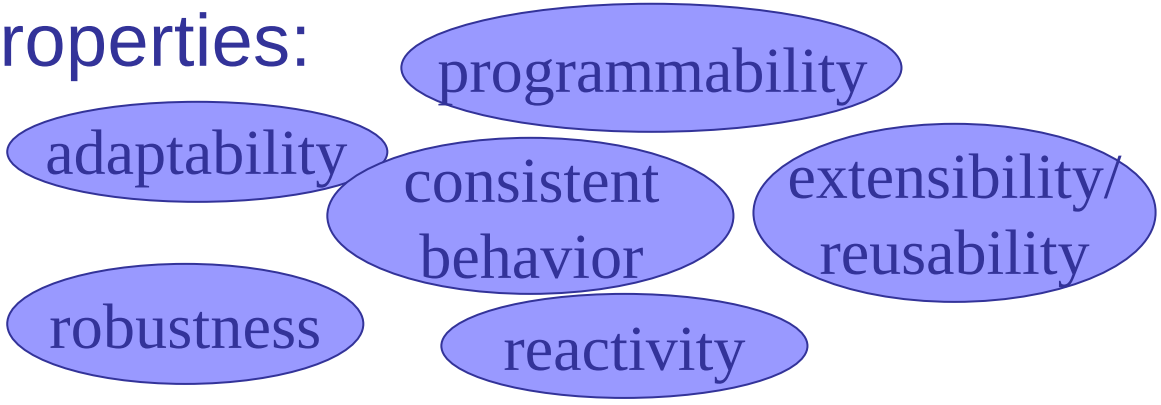
(www.laas.fr)

7 avenue du Colonel Roche

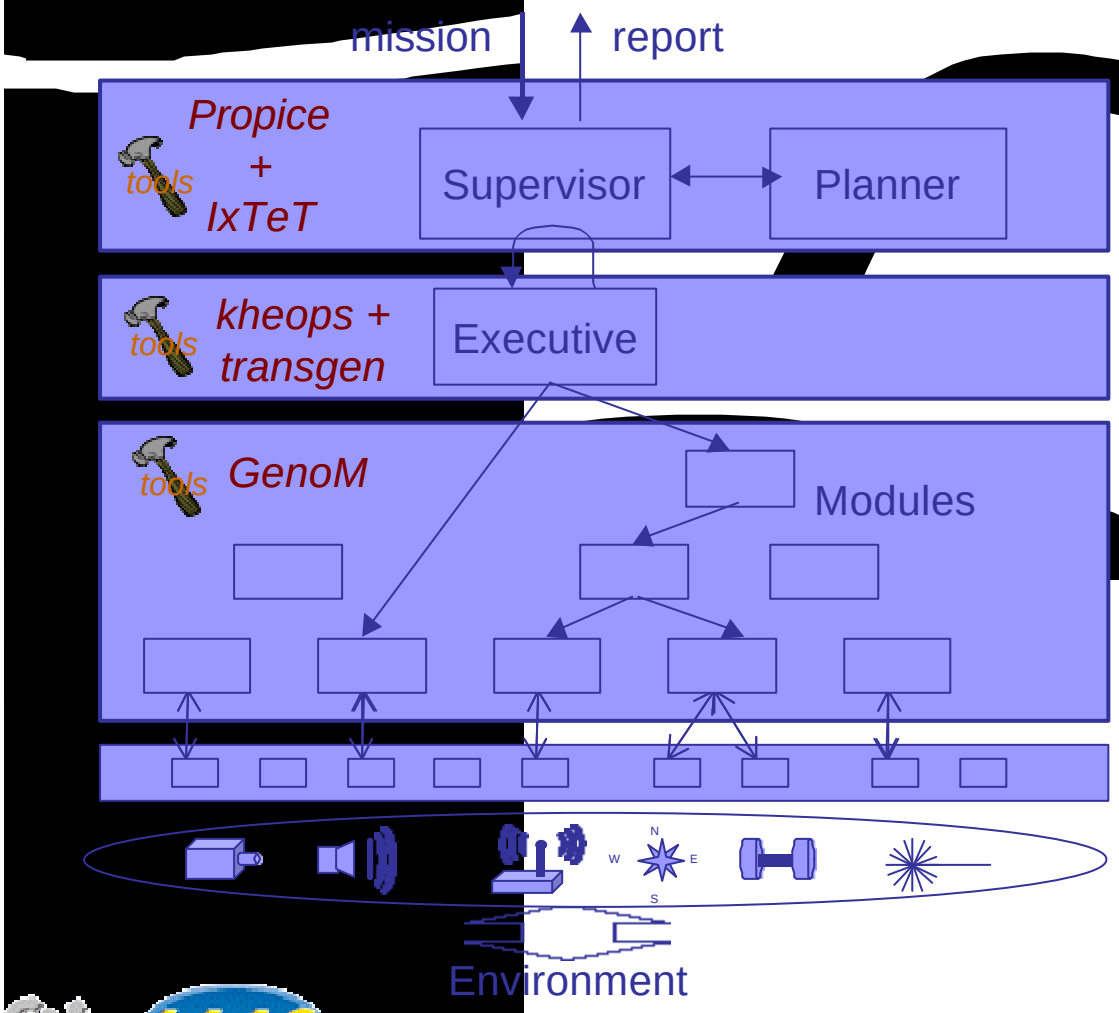
F-31077 Toulouse FRANCE

Motivations

- **Methodologies** and **tools** to design embeded software architectures for autonomous systems
- Results taken from **robotic research**
- Application to new generation of **satellites**:
  on board execution control and mission management
  - ground station maintenance simplified
  - flexibility and high level interactions
- Autonomy => reactive + decision making capabilities
- Architecture properties:



The 3 levels LAAS architecture: from decision to action



1. Decision Level
   (planning and supervision of action)
2. Execution Control Level
   (actions coordination)
3. Functional Level
   (*actions execution)*

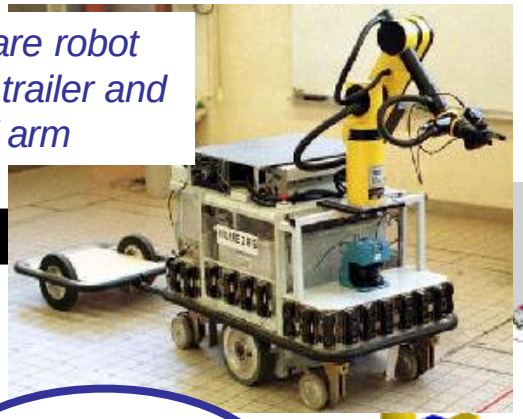
Logical System

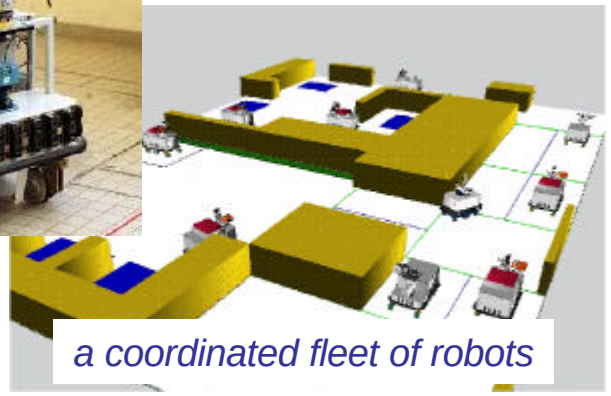
Physical Platform

From autonomous mobile robots ...

Diligent at an exhibition

an Hilare robot with a trailer and a 6dof arm





a coordinated fleet of robots

Service robotics

Planetary exploration robotics

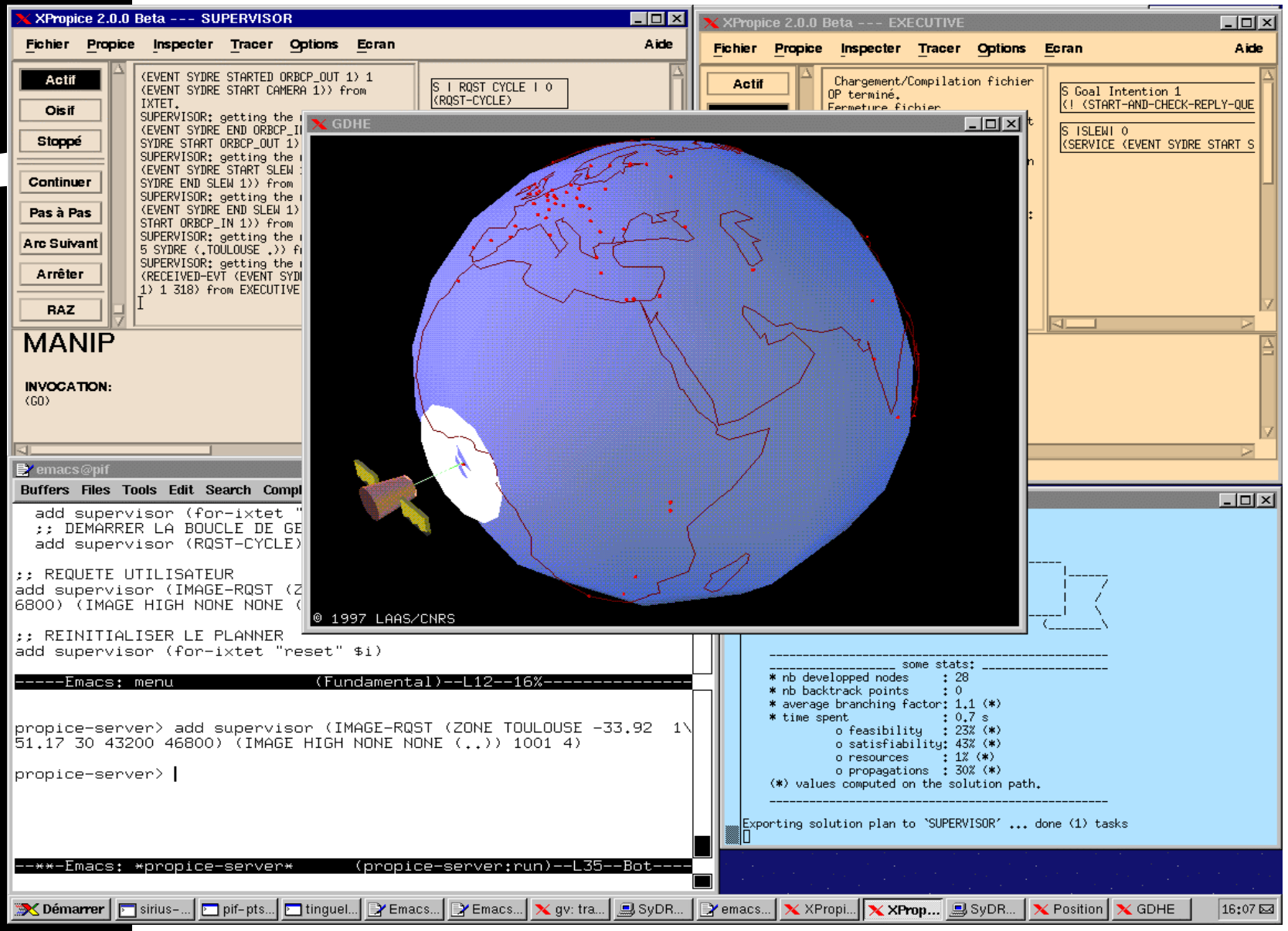
Trans-shipment robotics

the rover Lama

SNCF robot Commutor

new harbour of Rotterdam

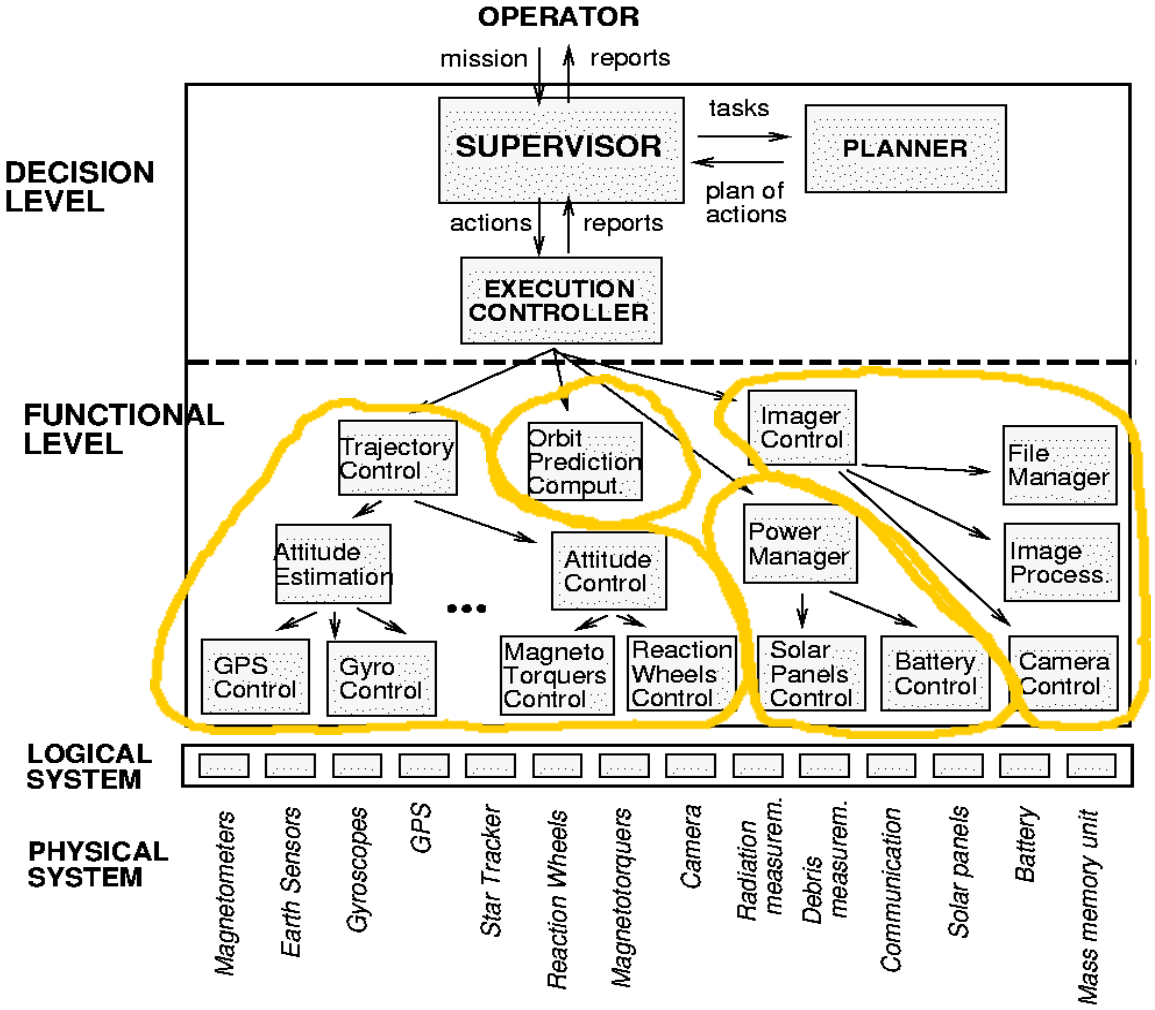
… to autonomous satellites



LAAS architecture for an autonomous earth observation satellite

Security/redundancy

- 1 module per sensor-actuator
- hierarchical modules organization in 4 sub-systems:
  - trajectory control
  - orbit prediction
  - power management
  - imager control

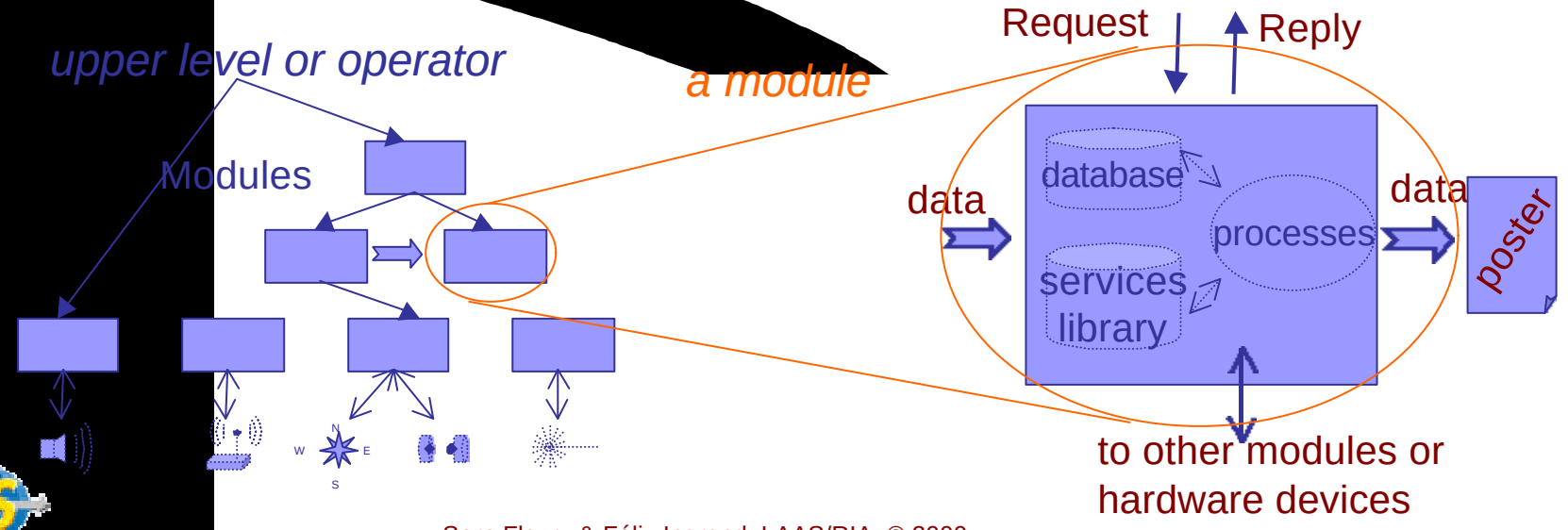


LAAS level 1: the Functional Level

Integrates all the operational functions
(hardware control, servo-control, data processing, …)

Structured as a set of independent **modules** (dynamically controlled by the upper level)

Module: entity responsible for a physical or logical resource



The Generator of Modules GenoM

- Automatic code synthesis
- No need to know the underlying OS
- One can concentrate on the functionalities
- Incremental design

1. module description

```
module Motion {
  number:  9600;
  SDI:     MOTION_DATA;
}
request SetPos {
  type:    control;
  input:   pos: :pos;
  control: controlPos;
  report:  BAD_PARAM;
}
task Move {
  period:  25;
  priority: 15;
}
```

2. module generation

GenoM

- parser
- generic module instantiation
- compilation
- link editing

executable module
(various OS)

Interface libraries
(C, Propice, TCL, ...)

test programs

```
STATUS
controlPos (POS_STR *pos, REPORT...)
{  if (pos->x < 0.0) {
     *report = BAD_PARAM;
     return ERROR; }
   return OK;
}
```

3. algorithms integration

4. tests

LAAS

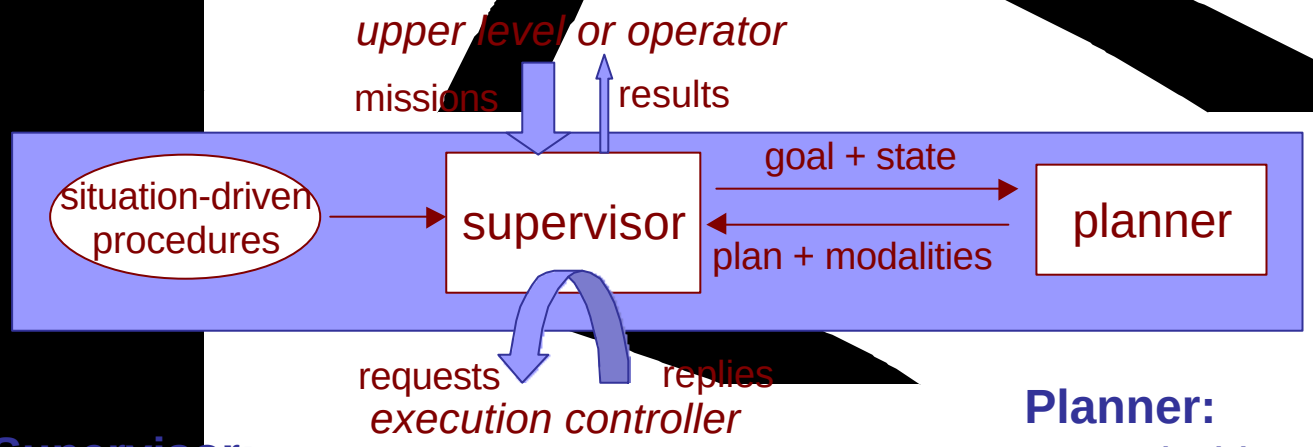
LAAS level 2: the Execution Control level

- **Pivot** between functional/decision levels
- Purely **reactive system** that reacts to decision level requests and functional level replies
- **State controller** of function level:
  - maintains functional level state
  - filters decision level requests
  - detects and manages conflicts
  - recovers failures locally
- 

kheops : automatic **automaton synthesis** from a set of propositional rules (complete, consistent, optimised)

LAAS level 3: the Decision Level

- All processes that require anticipation and global knowledge of the task and of the execution context.
- Structured in **supervisor-planner** layers:



Supervisor:
- Interprets upper mission
- Selects action procedures (or call planner)
- Controls the procedures execution
- Reacts to events (replies) from lower level
- tool : PROPICE

Planner:
- Queried by supervisor
- Deals with:
  - time constraints
  - resources constraints
  - predictable events
- Produces plan of actions
- tool : IxTeT

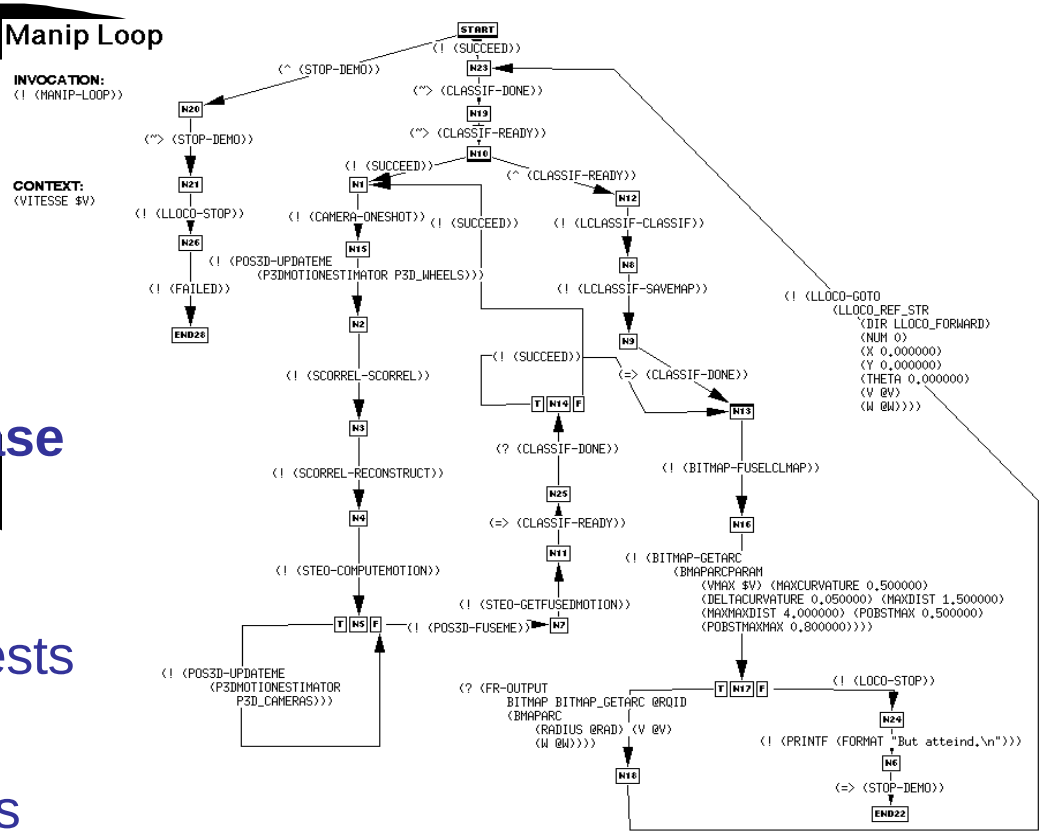
The Procedural Reasoning System PROPICE

Properties:

- high-level language
- parallel tasks + asynchronous events handling
- temporal properties

Main components:

- automatically updated **database** (view of the world)
- a library of **procedures**:
  - sequence of actions and tests
  - to achieve given goals, or
  - to react to certain situations
- a dynamic task graph



Example of a PROPICE procedure

IxTeT Temporal Planner

IxTeT: IndeXed Time Table

- IxTeT kernel: an efficient time-map manager
- Time-point algebra relations and restricted interval algebra
- Used in situation recognition and plan synthesis
- Common knowledge representation: **chronicles**

Example of an IxTeT plan