Orocos @ LAAS

One year after...

Anthony Mallet, Sara Fleury, Raja Chatila

LAAS - CNRS

mallet@laas.fr, sara@laas.fr, raja@laas.fr

Leuven, November 18th, 2002

Outline

- **Software Architecture**
  - Approach overview and existing tools.
- **Salient Examples**
  - Robots, operating systems
- **Orocos@LAAS**
  - Last year work

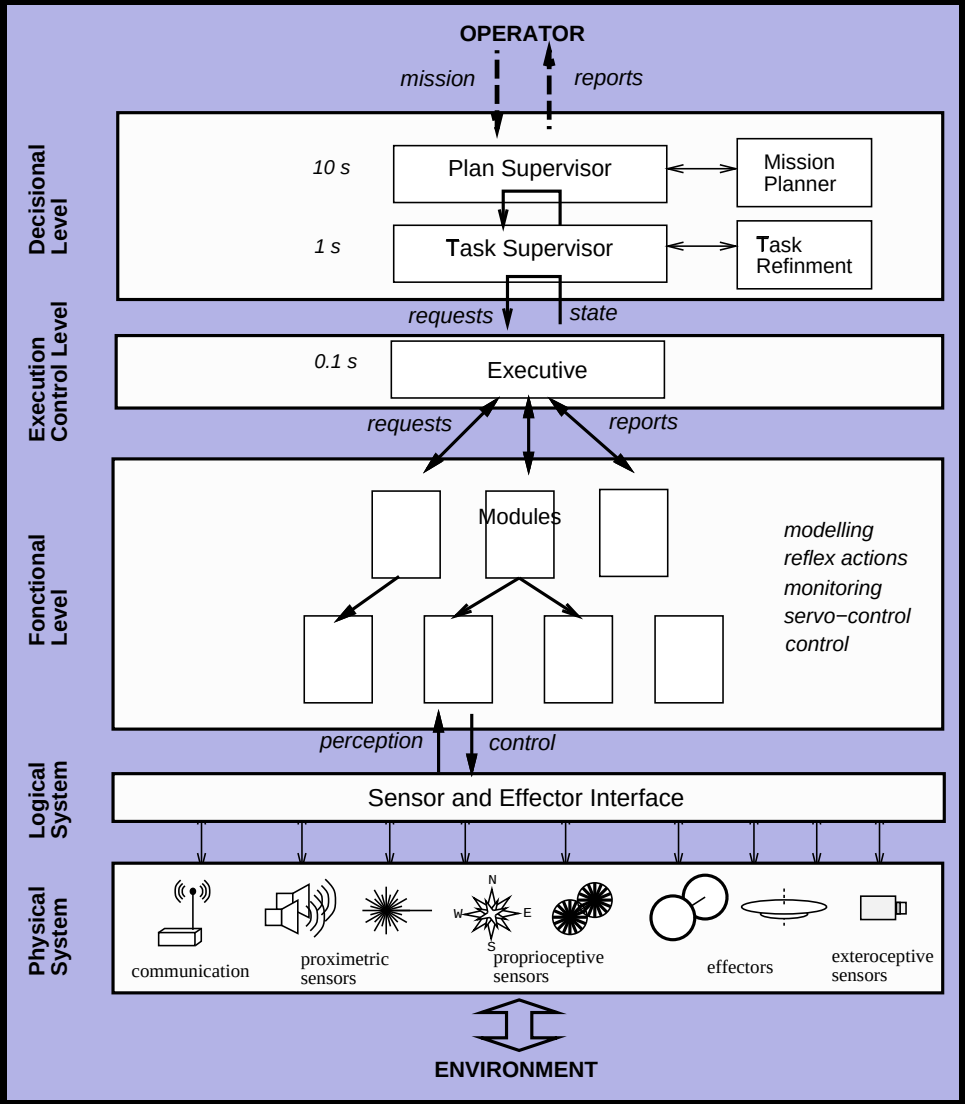
Context: Autonomous Robots

Systems and software are becoming more and more complex.

- many functionalities
- decisional aspects

- A real-time distributed system
  - Heteregenous functionalities
  - Inter-task communication
- A supervised system
  - Dynamic control
  - Dynamic parametrization
  - Observability
- An open system
  - Incremental design

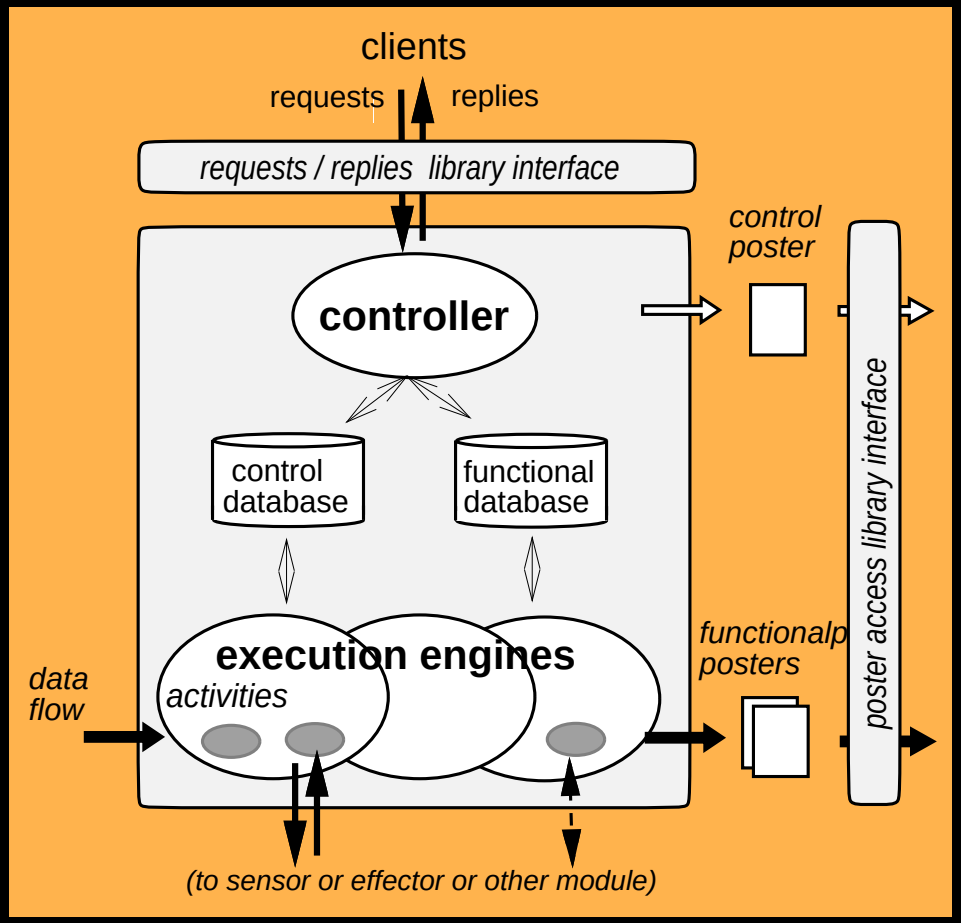
Strong need for software architectures


OPERATOR
mission
reports
Decisional Level
10 s
Plan Supervisor
Mission Planner
1 s
Task Supervisor
Task Refinment
requests
state
Execution Control Level
0.1 s
Executive
requests
reports
Fonctional Level
Modules
modelling
reflex actions
monitoring
servo−control
control
perception
control
Logical System
Sensor and Effector Interface
Physical System
N
W
E
S
communication
proximetric sensors
proprioceptive sensors
effectors
exteroceptive sensors
ENVIRONMENT

Functional Level: Generic Components

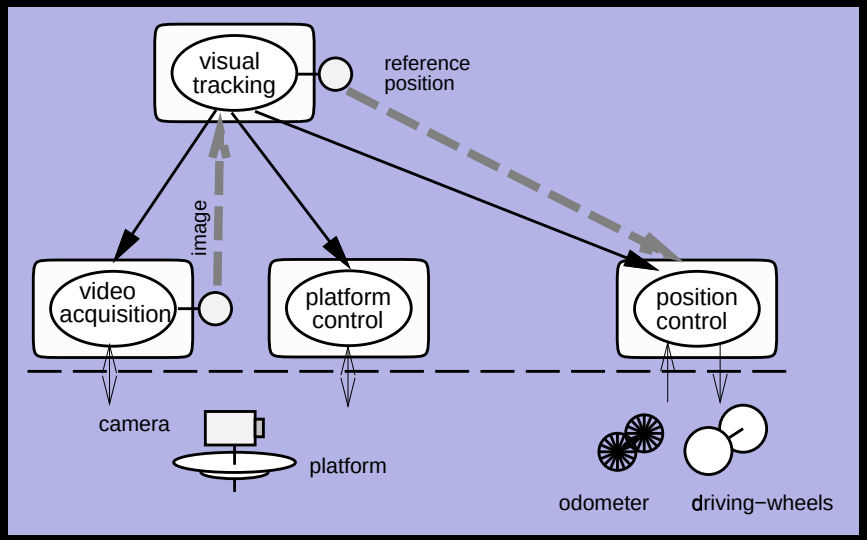
Component: entity offering *services* that endow the functional layer with a particular *capability*.

Services are parametrized and started by asynchronous *requests*, which in turn spawns activities whithin the *execution engines*.



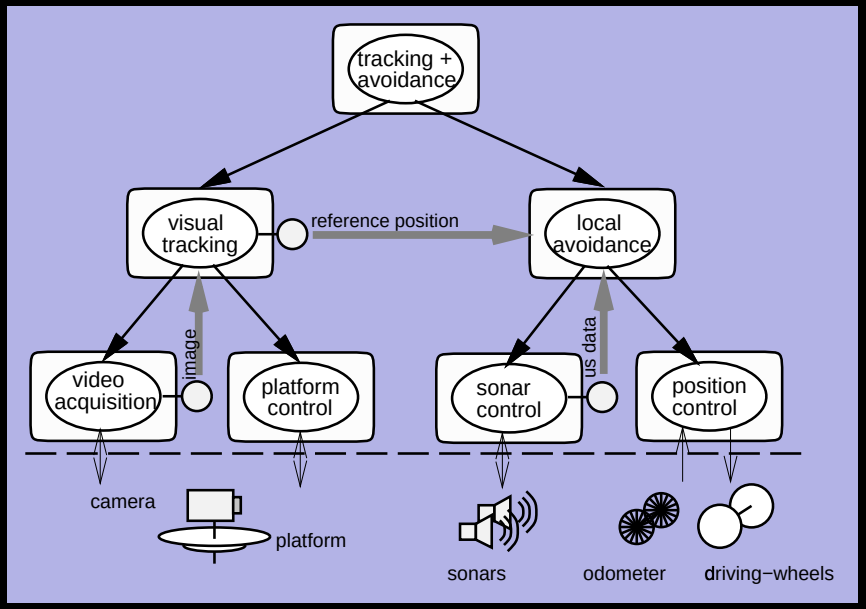
1. visual tracking

uses 3 basic services



2. visual tracking + obstacle avoidance

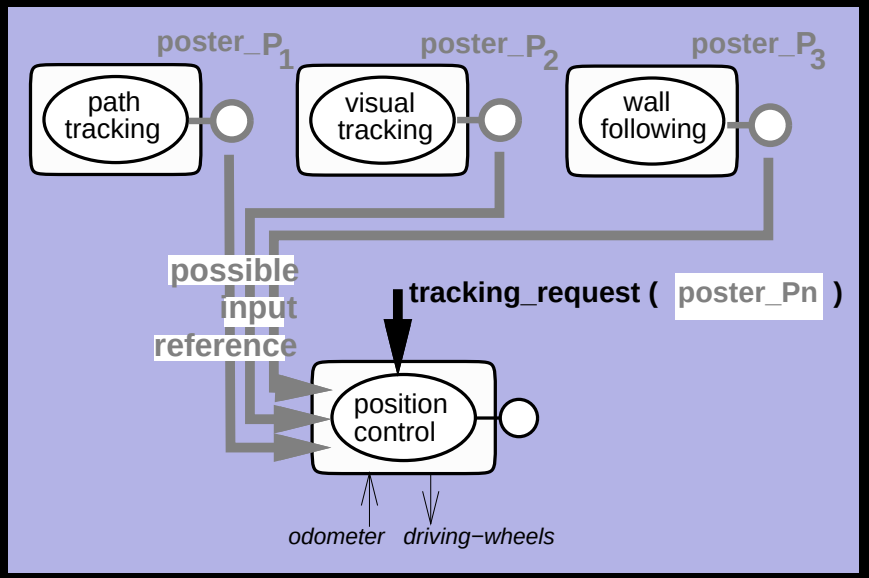
7 modules are now involved



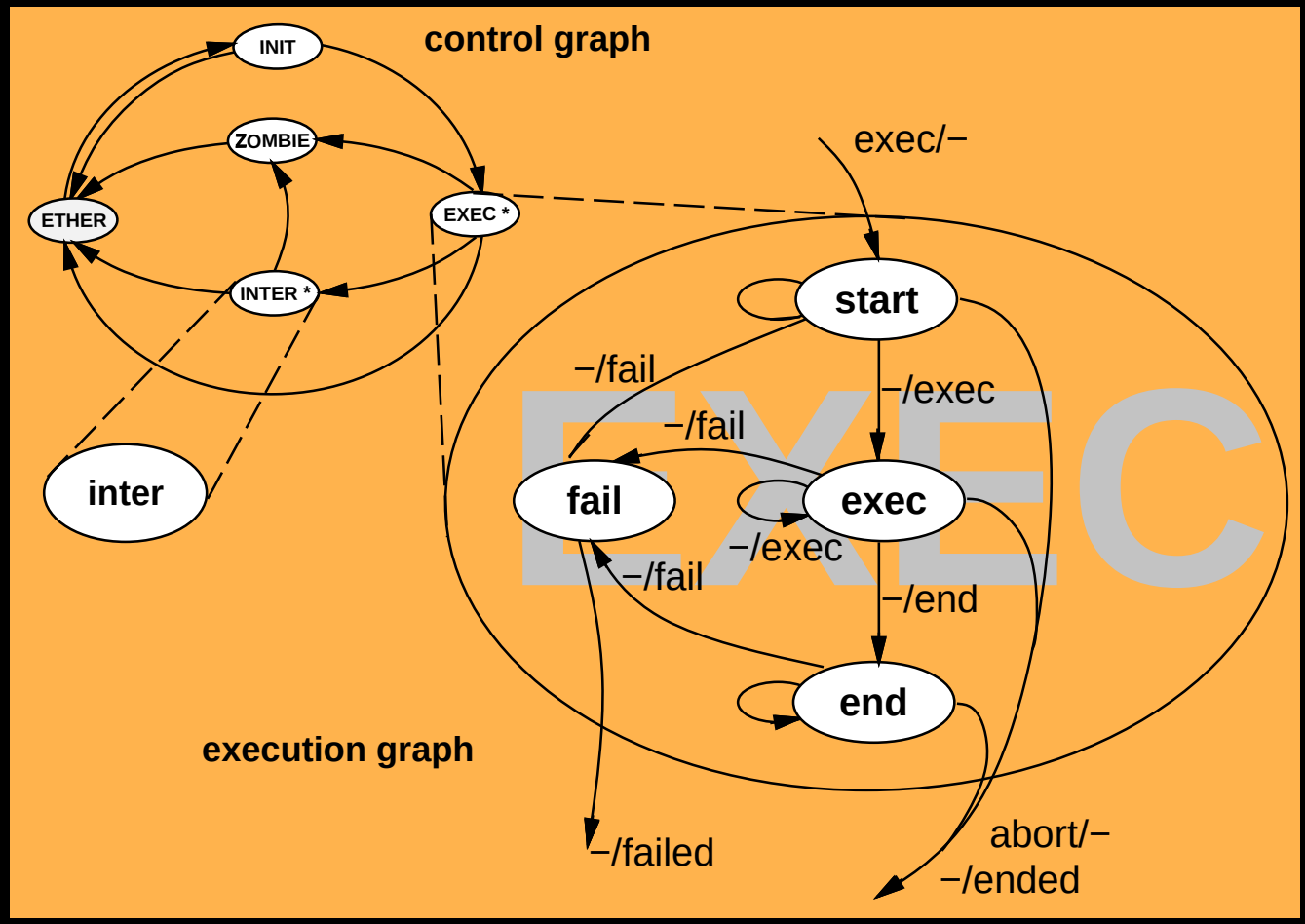
Example: Three possible sources for position control

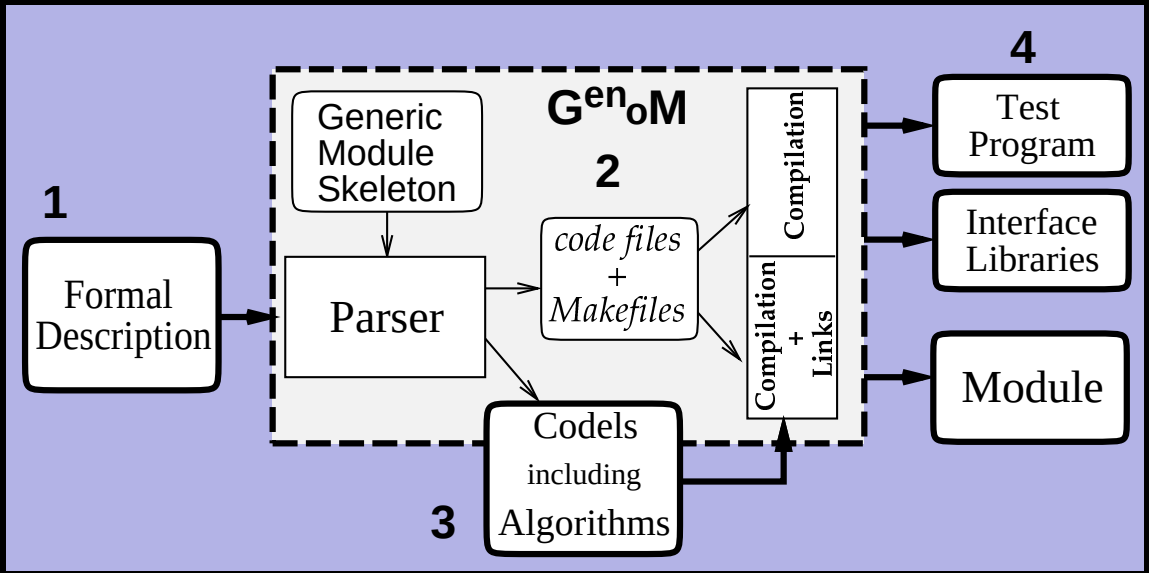
Poster

- Structured shared memory
- Writable only by the owner (component that export the data).
- Readable by any other component
- Allow data flow redirection



Activity state diagram

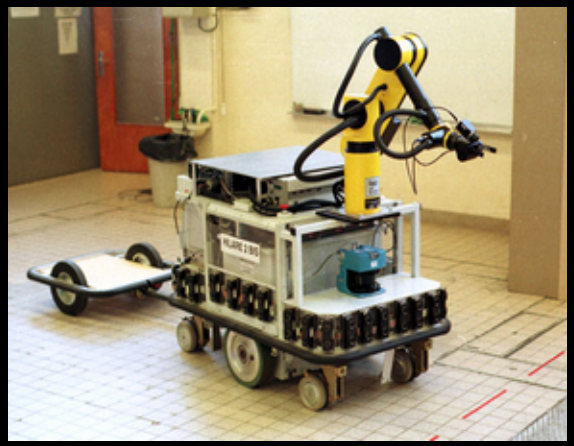


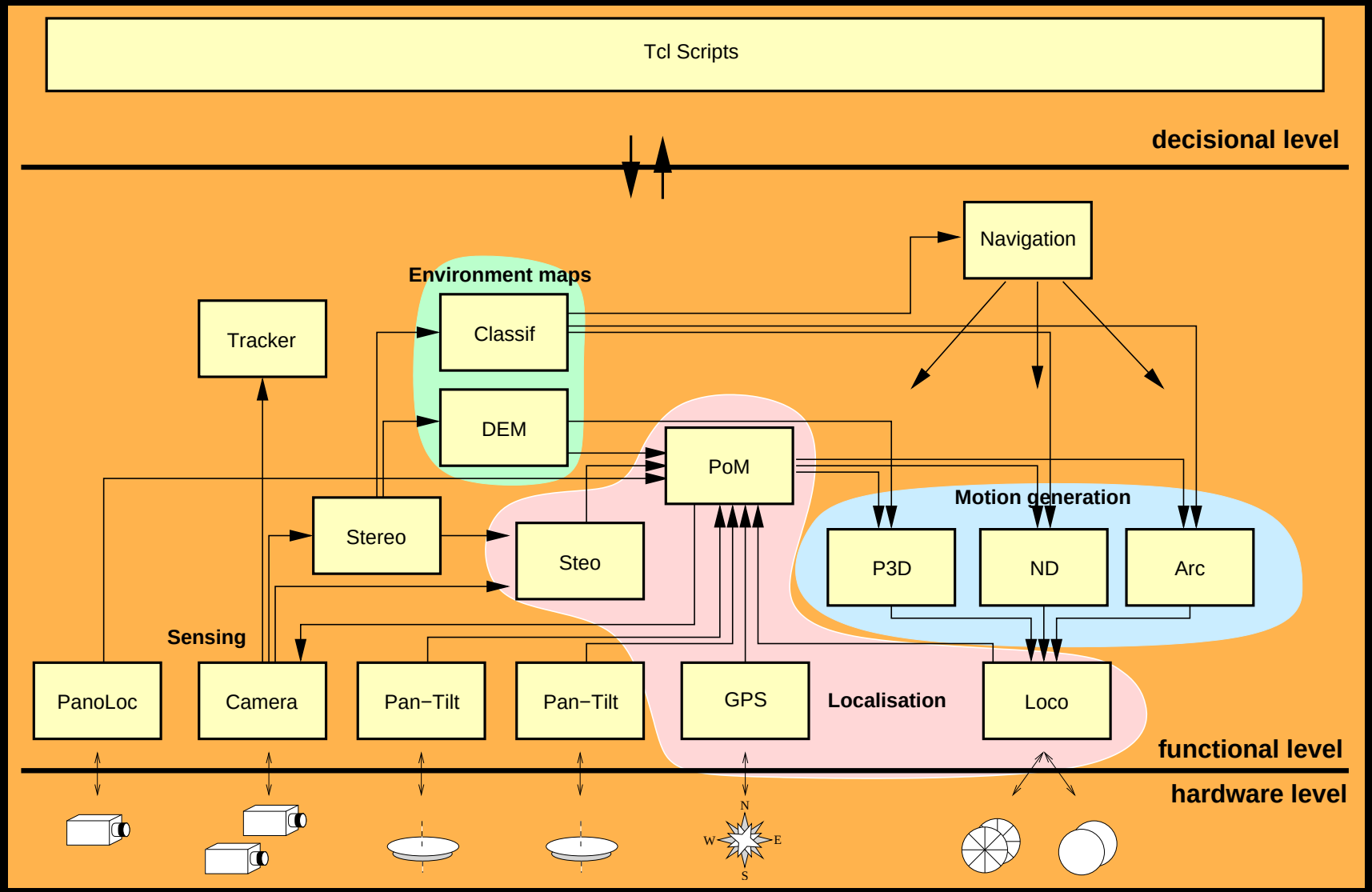


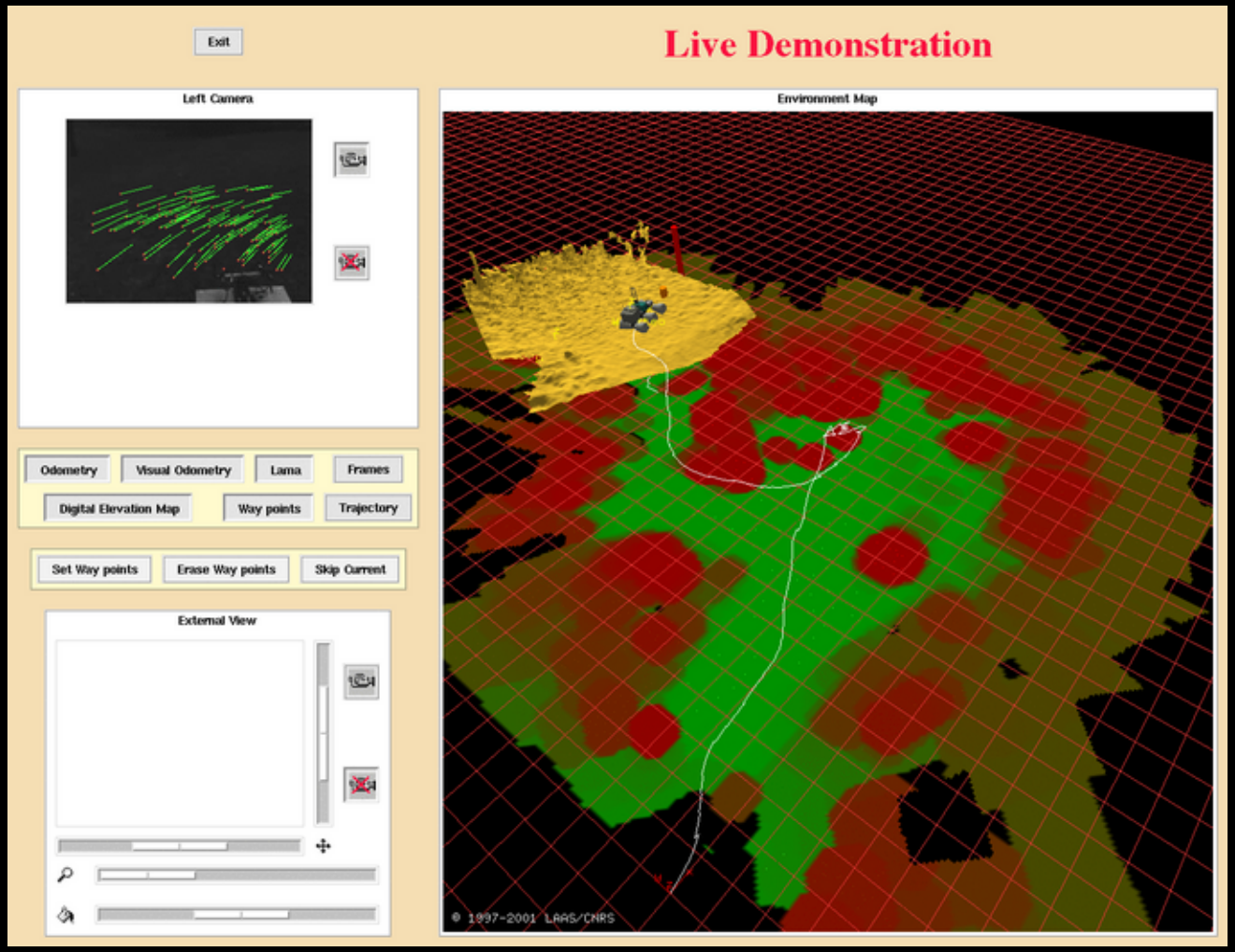
- **Formal description**: input for GenoM
- **Generation of the component**: code generation and compilation
- **Test and validation** programs.

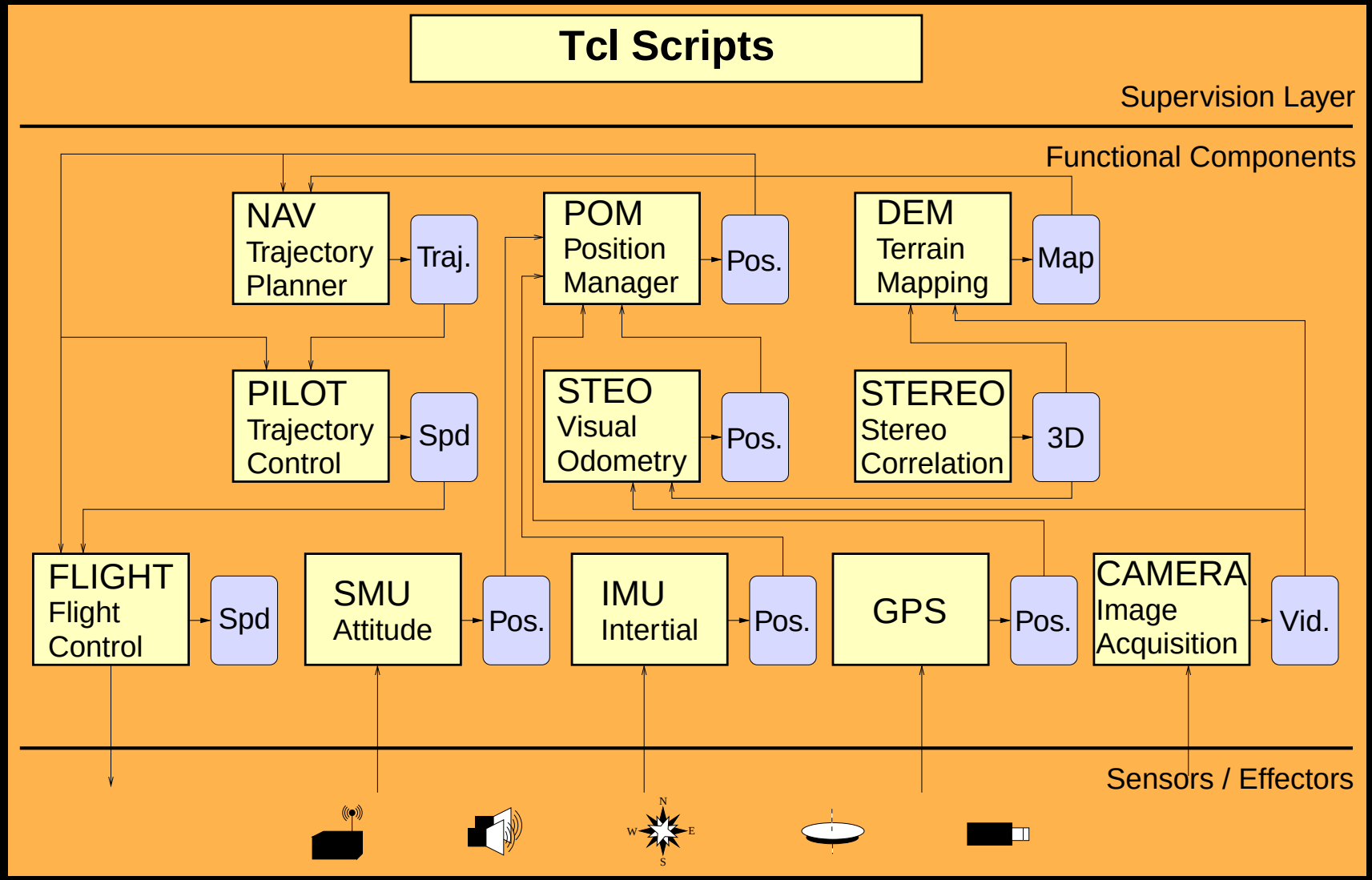
Salient Examples

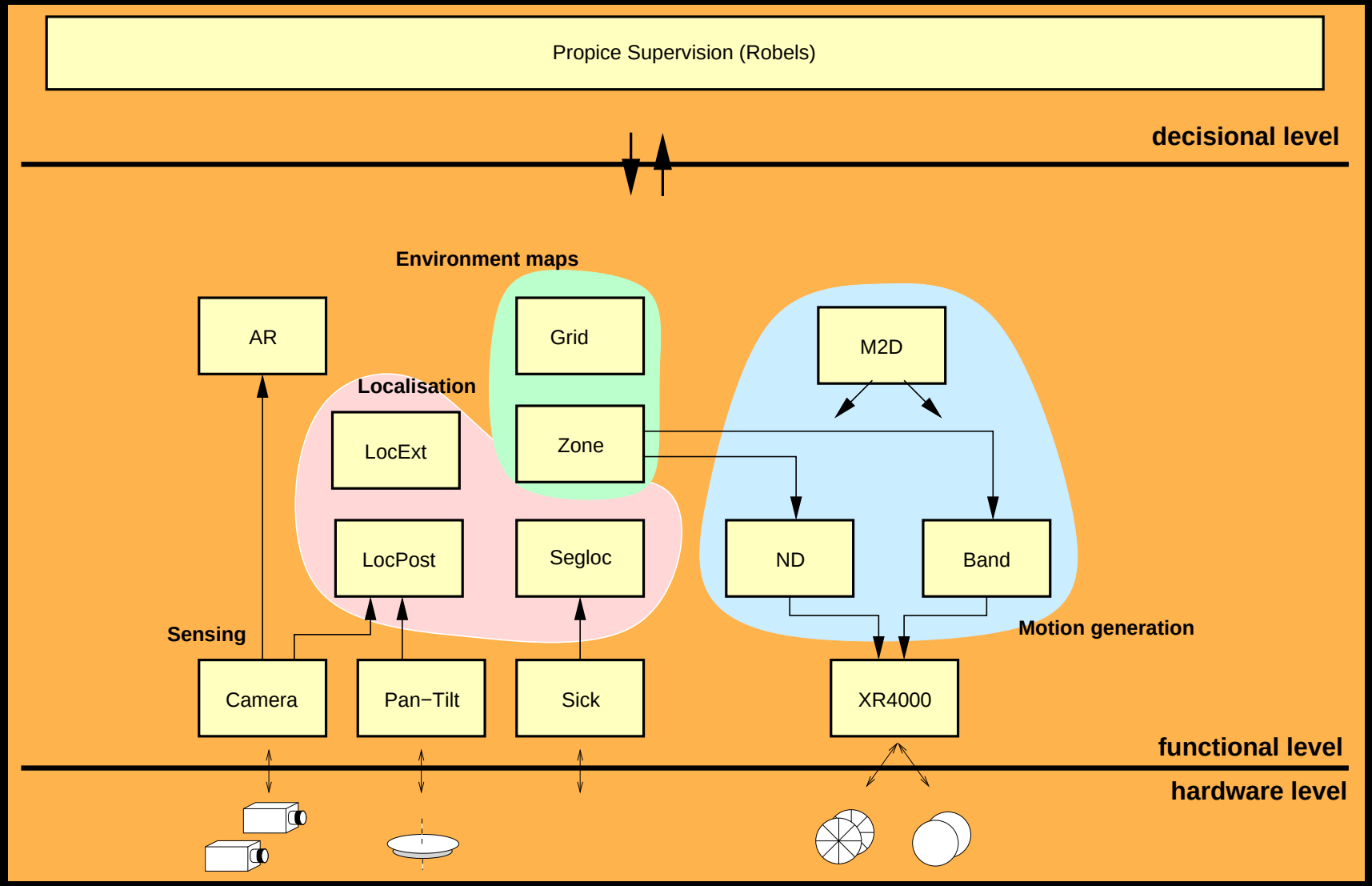
Robots




Tcl Scripts
decisional level
Navigation
Environment maps
Tracker
Classif
DEM
PoM
Motion generation
Stereo
Steo
P3D
ND
Arc
Sensing
PanoLoc
Camera
Pan−Tilt
Pan−Tilt
GPS
Localisation
Loco
functional level
hardware level
N
W
E
S


Exit
Live Demonstration
Left Camera
Environment Map
Odometry
Visual Odometry
Lama
Frames
Digital Elevation Map
Way points
Trajectory
Set Way points
Erase Way points
Skip Current
External View
© 1997-2001 LAAS/CNRS


Tcl Scripts
Supervision Layer
Functional Components
NAV
Trajectory
Planner
Traj.
POM
Position
Manager
Pos.
DEM
Terrain
Mapping
Map
PILOT
Trajectory
Control
Spd
STEO
Visual
Odometry
Pos.
STEREO
Stereo
Correlation
3D
FLIGHT
Flight
Control
Spd
SMU
Attitude
Pos.
IMU
Intertial
Pos.
GPS
Pos.
CAMERA
Image
Acquisition
Vid.
Sensors / Effectors
N
W
E
S


Propice Supervision (Robels)
decisional level
Environment maps
AR
Grid
M2D
Localisation
LocExt
Zone
LocPost
Segloc
ND
Band
Sensing
Motion generation
Camera
Pan−Tilt
Sick
XR4000
functional level
hardware level

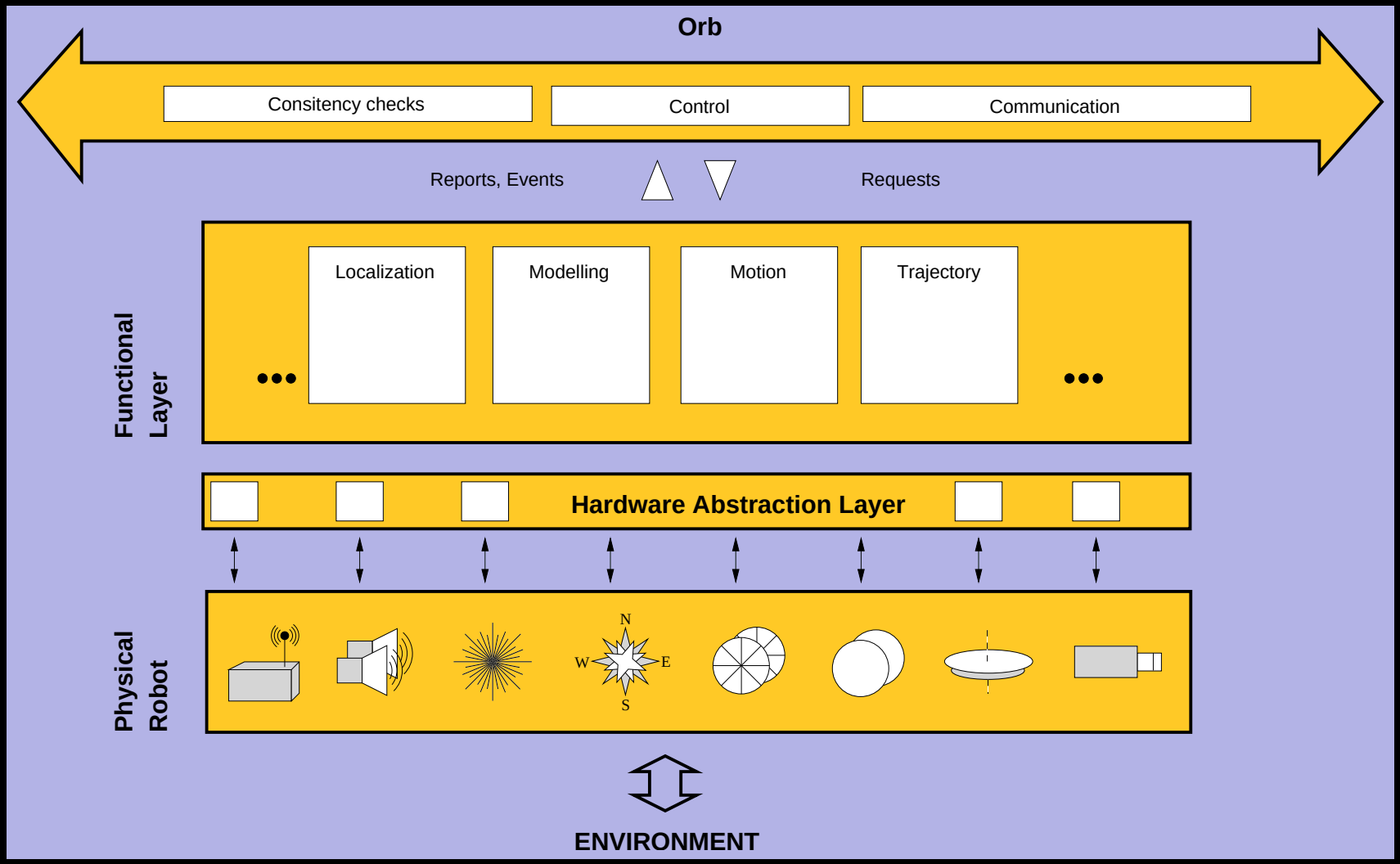
Development of a **software framework** that will:

- Maximize **reusability** of software
- Allow **software and application sharing** between labs and researchers
- Achieve a **great modularity**; allow **dynamic connection** of different components together
- Eventually define **a standard model** for robotic software components.

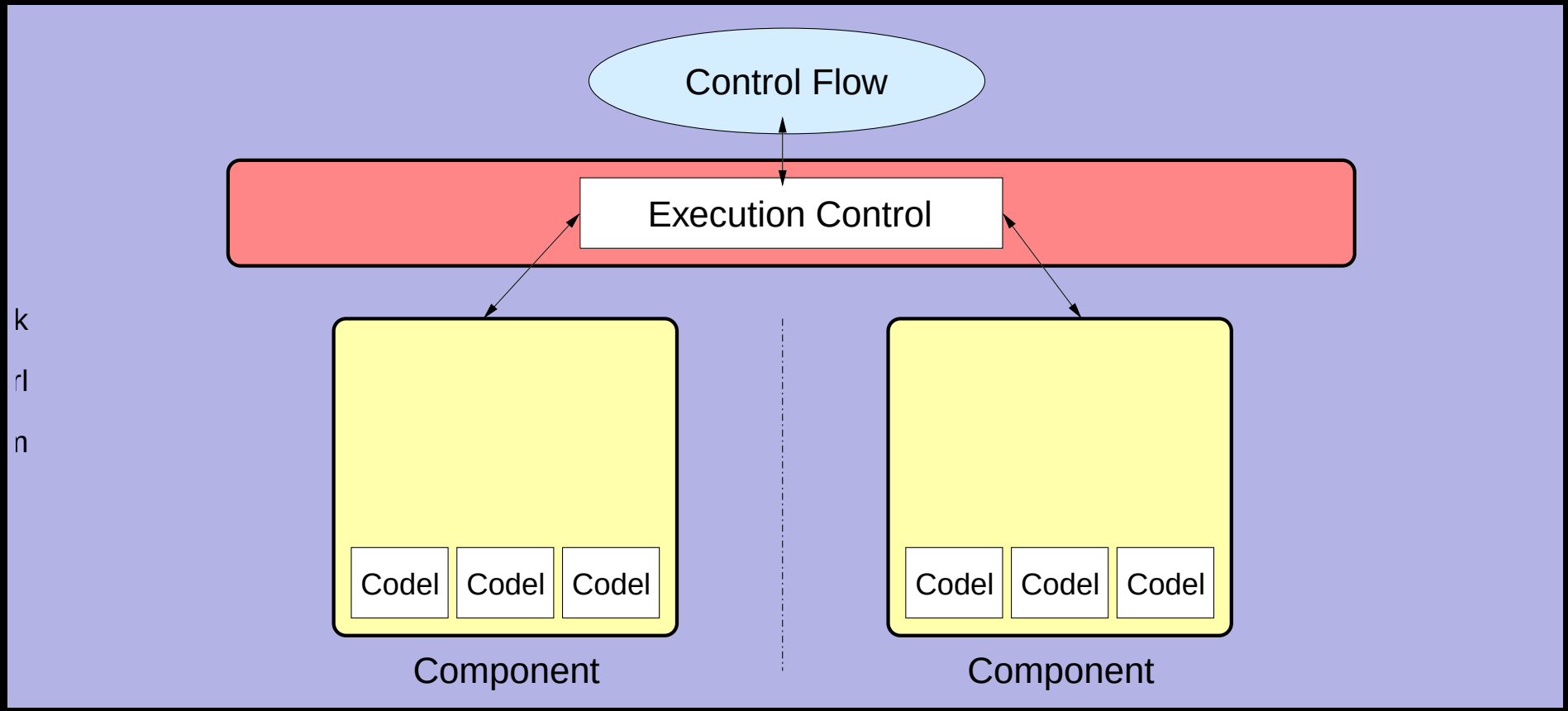
Software architecture != Decisional architecture

but software architecture must take into account the constraints raised by the different decisional architectures that have been proposed.

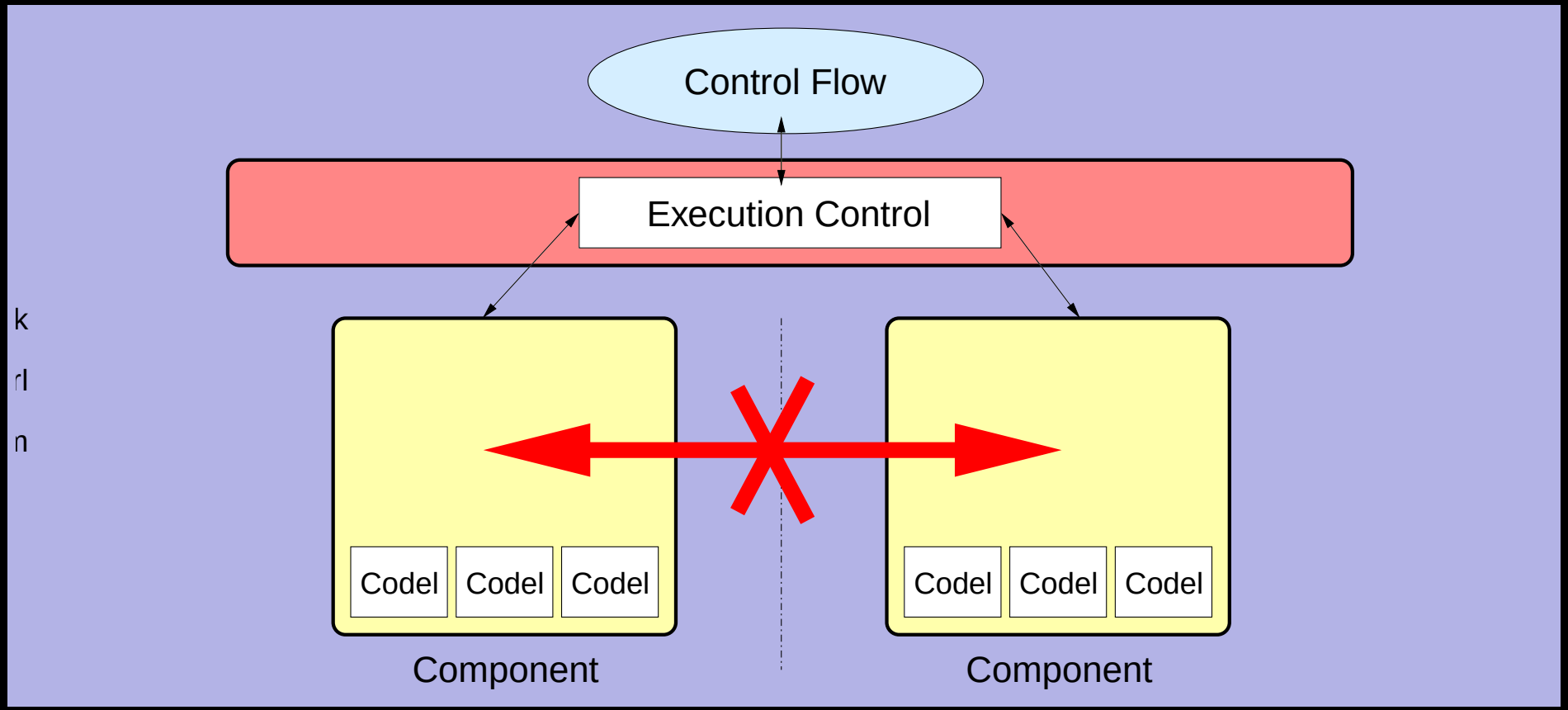
Functional layer



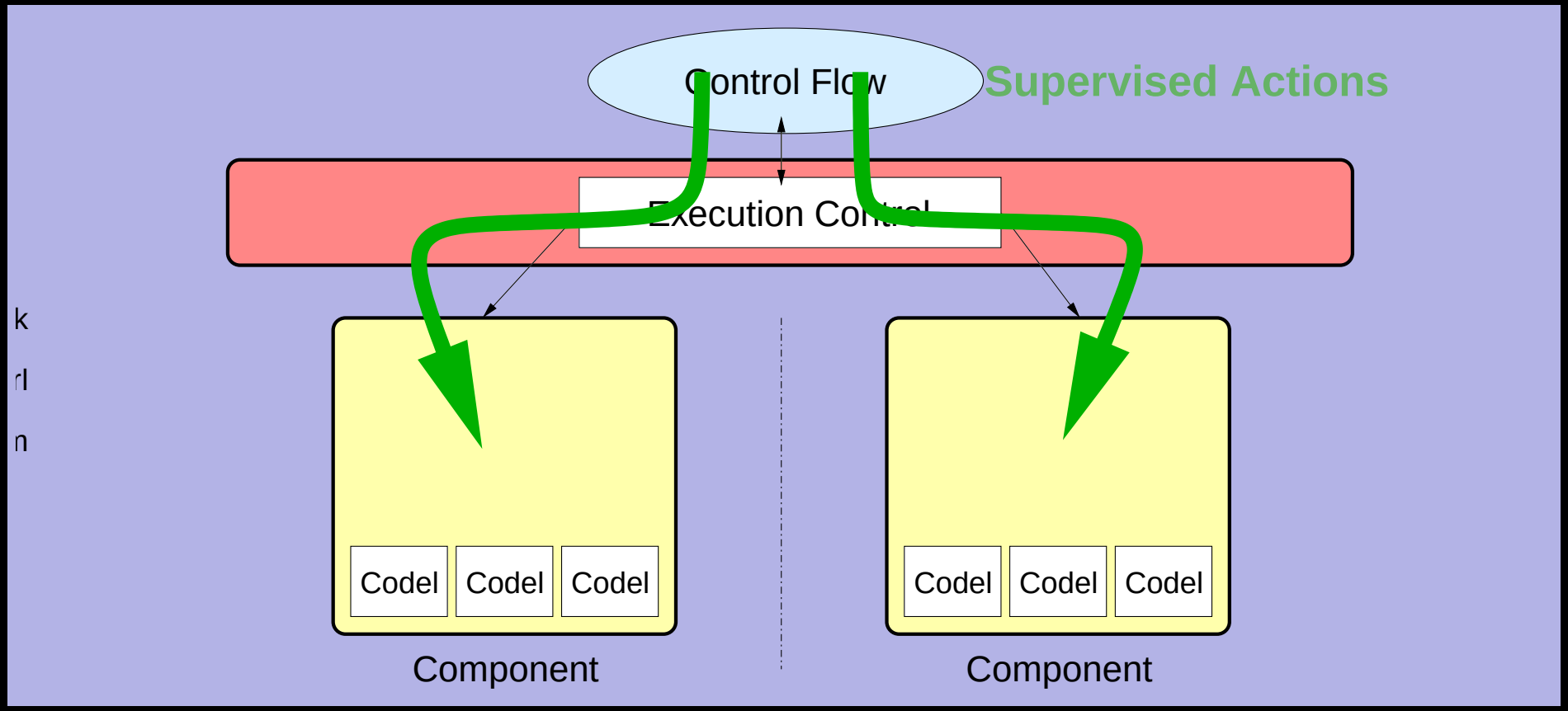
- The control flow has to be defined outside components.



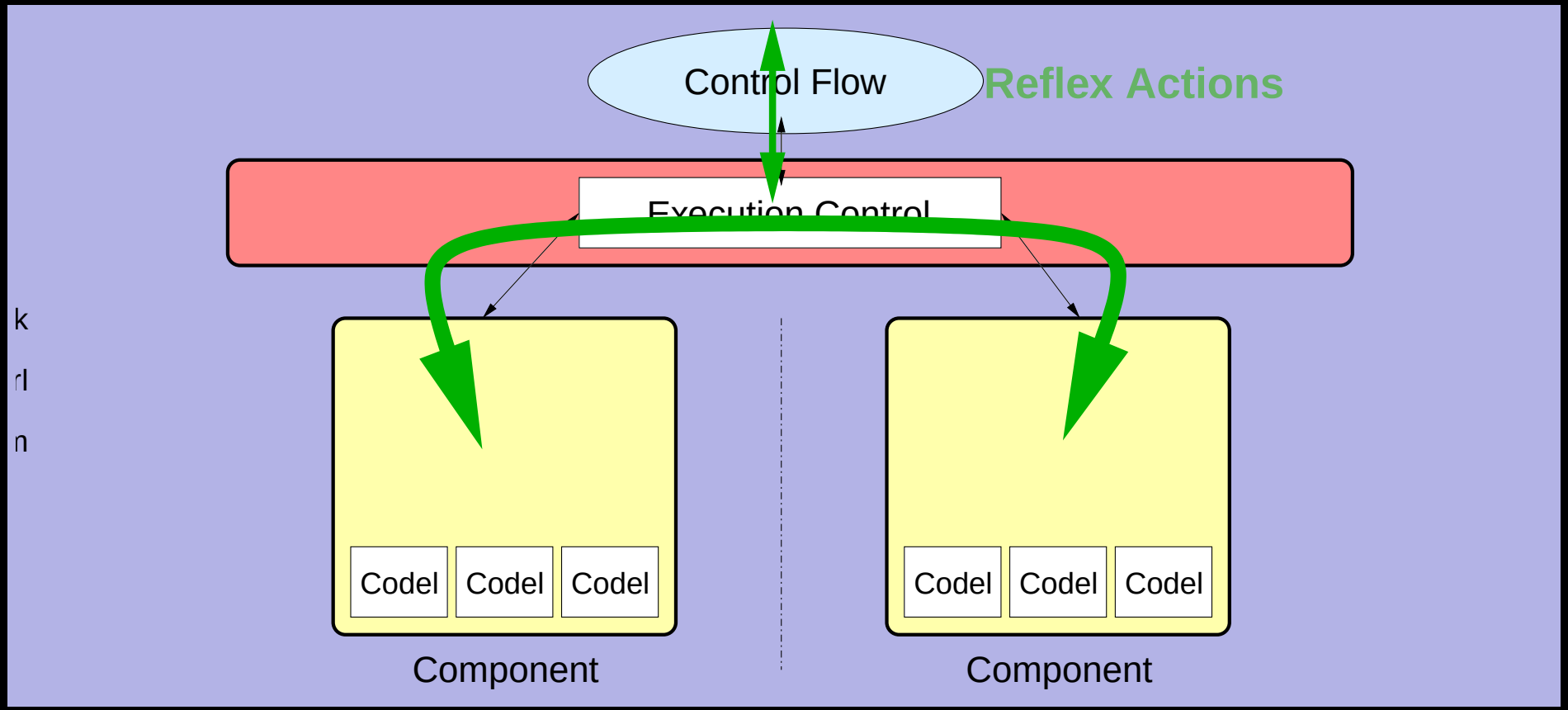
- The control flow has to be defined outside components.

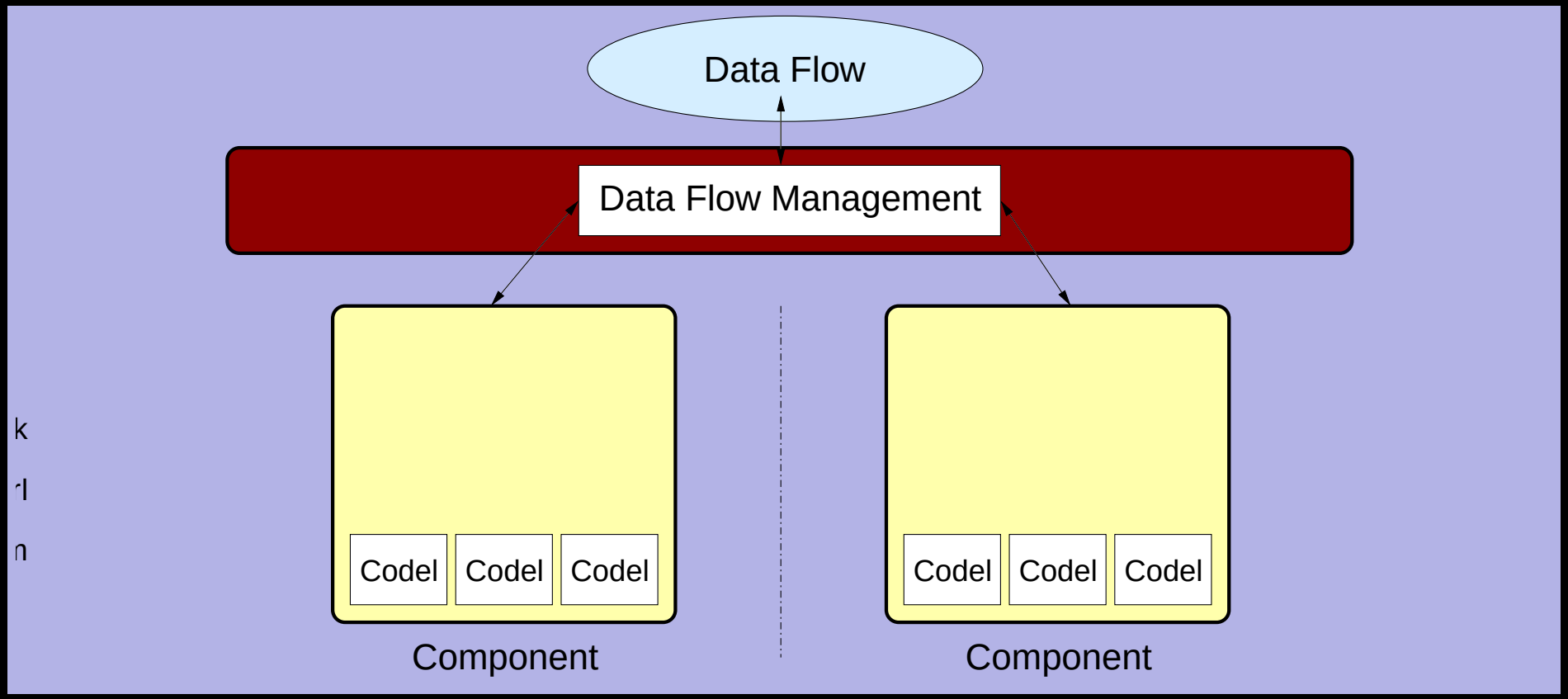


- The control flow has to be defined outside components.

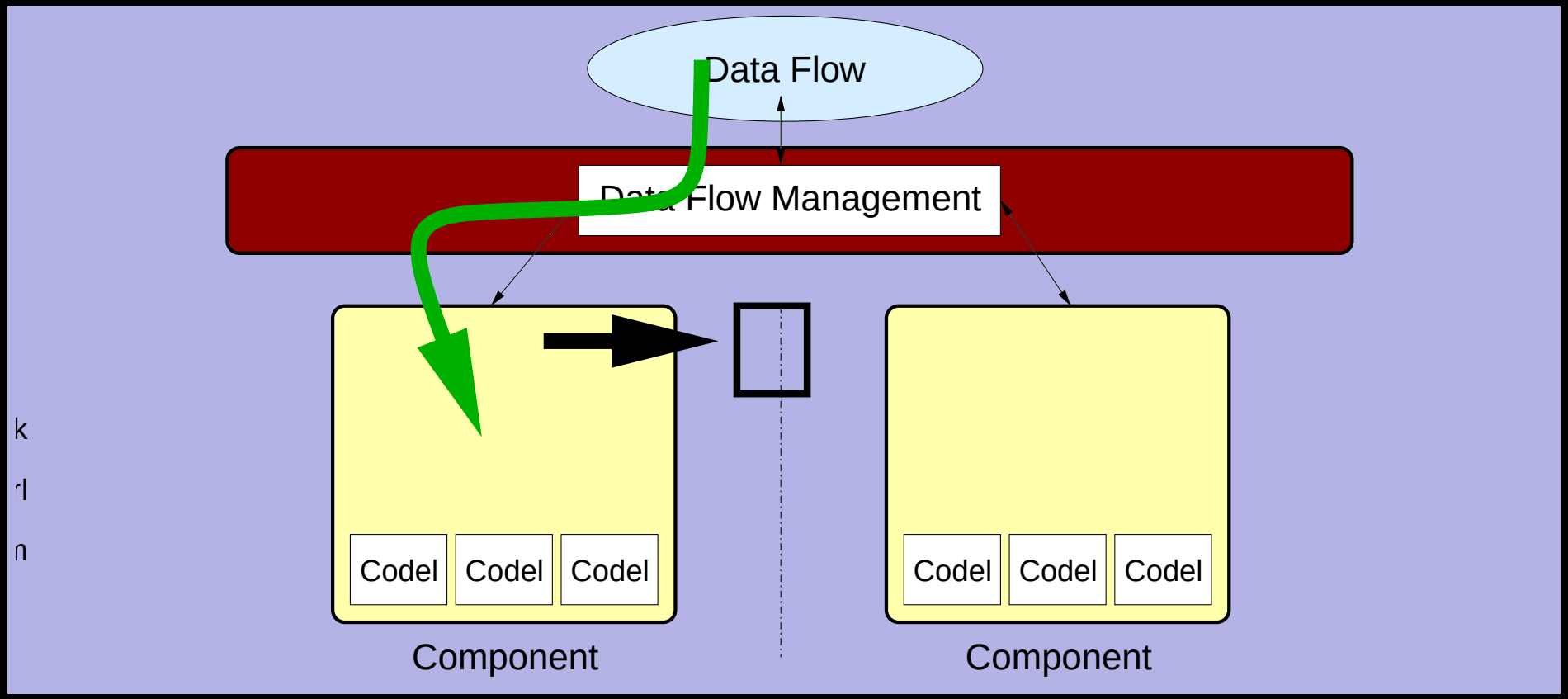


- The control flow has to be defined outside components.

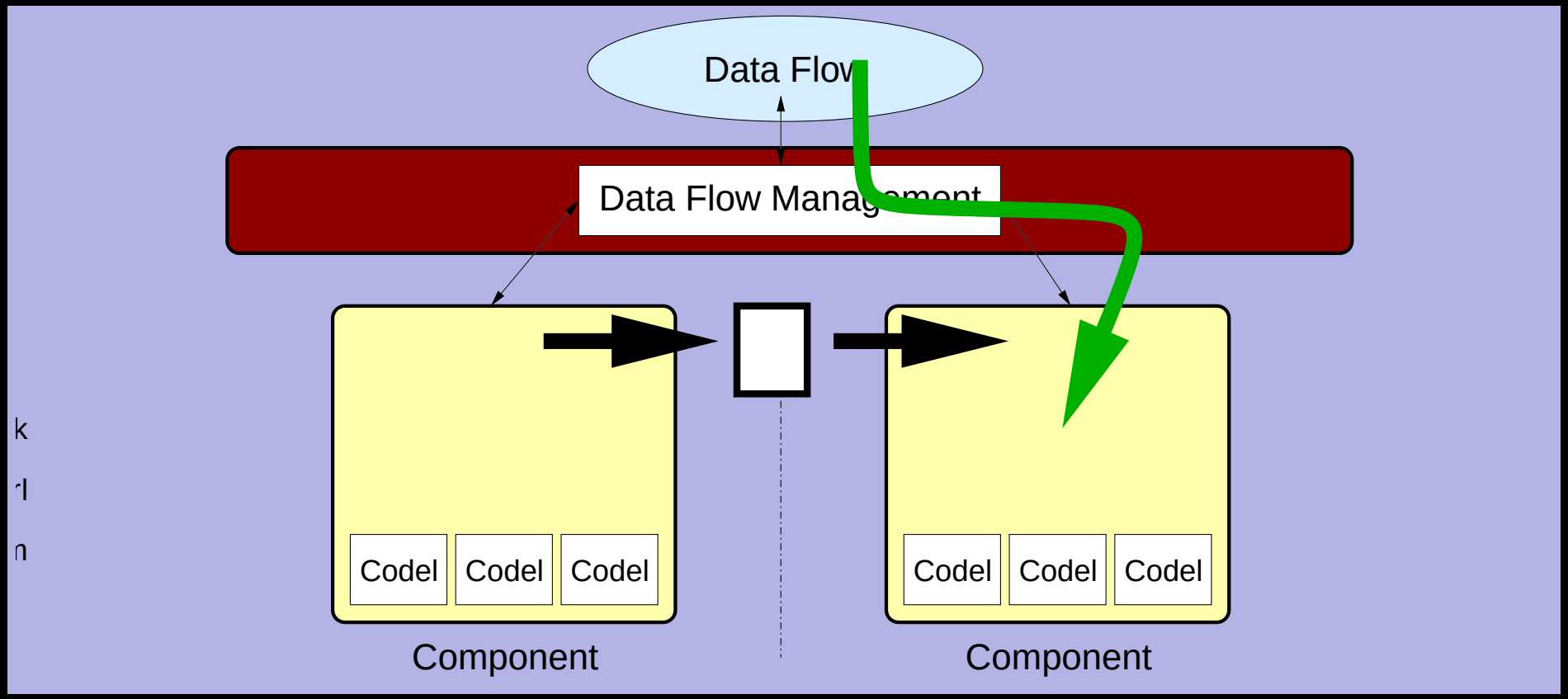




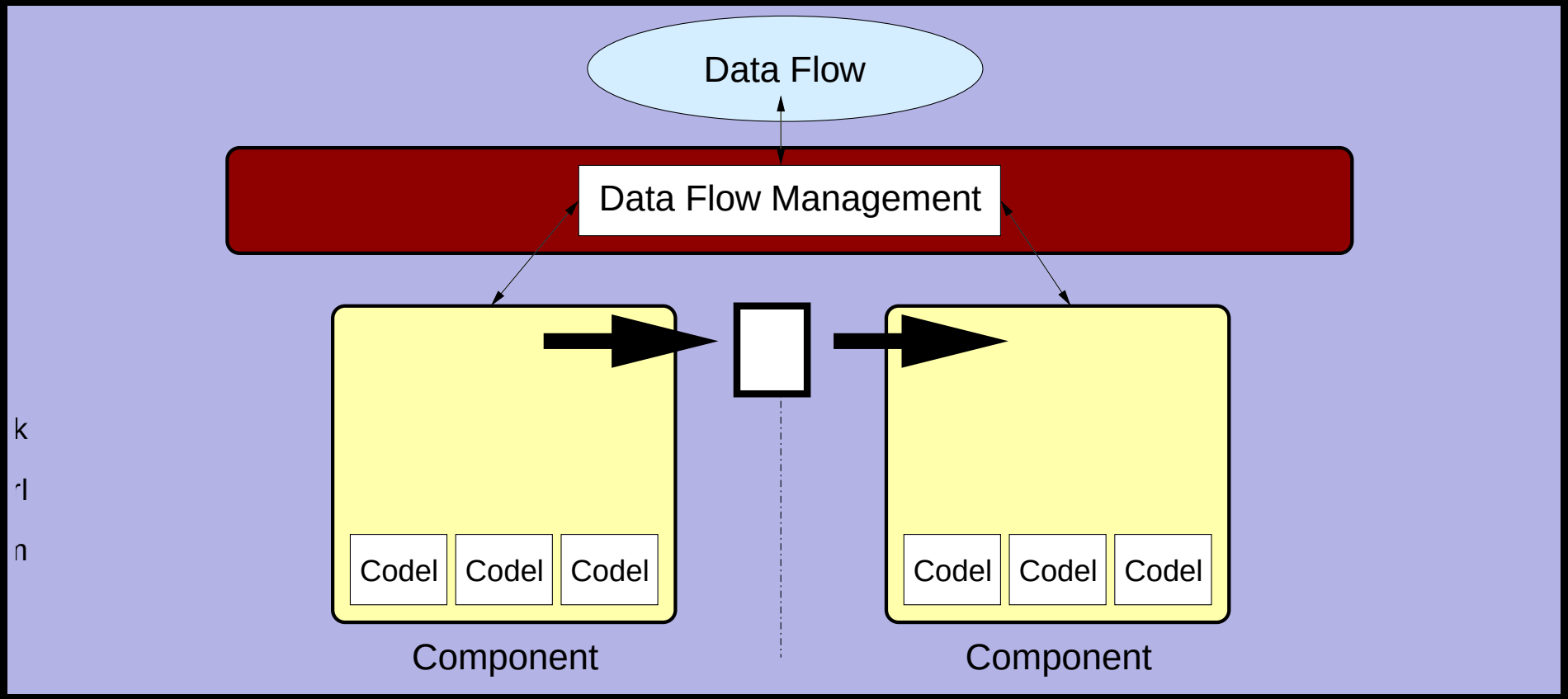
- The data flow has to be defined outside components too.



- The data flow has to be defined outside components too.



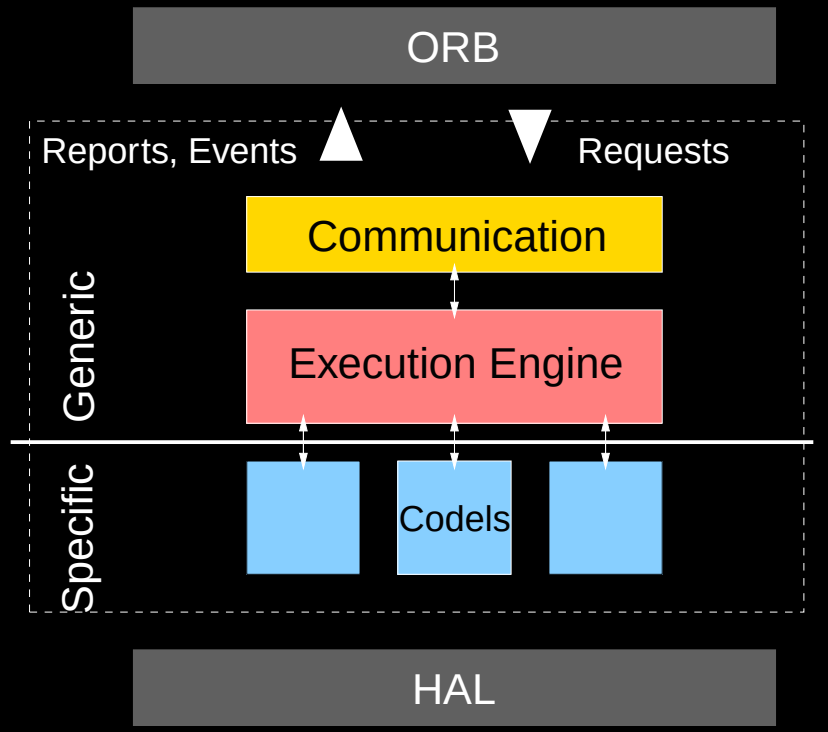
- The data flow has to be defined outside components too.



- The data flow has to be defined outside components too.

Components are made of three parts

- **Codels**: Actual code of the component, split into elementary states,
- **Execution engine**: FSM that executes the codels,
- **Communication**: import and export data and parameters.



Codels do not contain code related to communication or sequencing

- **The formal description of a component contains:**

  → the list of services,

  → the set of codels for each services,

  → the services and codels interfaces (imported / exported data, parameters, ...),

  → the data structures that define parameters and exported data,

  → other information not detailed here.

- **The formal description is a text file.**

  → Data structures might be described in IDL (idependant of the programming language used for the codels definition)

  → IDL is not sufficient.

Overview of a service description (inputs, outputs, exports, imports, codels, ...):

```
service <name> {
   doc: "short description of the service";

   thread: <name>;

   /* input/output parameters */
   input|output: <type>, [<type>, ...];

   /* imported/exported data */
   import|export: <type>, [<type>, ...];

   codel <name> {
      exec: <function>( [const] <variable>, ...);
      max-time: <seconds>;
      next: <codel> [, <codel>, ...];
   }

   ...
}
```

- **Orocos must define standard objects.**

  $\longrightarrow$ IDL is a good choice for this task.

- **Empty slide :-)**

[OROCOS] European project (2001-2004) — **EURON SIG**

Open robot control software

[COMETS] European project (2002-2005)

Real time coordination and control of multiple heterogeneous unmanned aerial vehicles

AICIA (Portugal), LIU (Sweeden), LAAS (France), TUB (Germany)

- Disaster monitoring, environmental surveillance.

[Nasa-Ames] — Transfert

Autonomous Navigation in Outdoor Environments

Felix.Ingrand@laas.fr

- ATRV Junior, Linux.

[ACFR] (Sydney) — in the process of being defined

- Software architecture

Ongoing and Future Work, Deadlines

- **New GenoM tool — version 2**
  - More decoupling, more programming languages, more 'standards', BSD license.
  - Version 2.0 in the beginning of 2003?
- **Integration of Orocos@KULeuven**
  - Generic motion control component and dedicated execution engine.
  - Version 0 in summer 2003?
- **Integration of Orocos@KTH**
  - CORBA-based communication. Links with Orocos@KULeuven.
  - 2003?
- **Data structures repository** $\longrightarrow$ 2003!