ORCCAD, *a framework for safe robot control design and implementation*

Daniel Simon, Roger Pissard-Gibollet and Soraya Arias

INRIA Rhône-Alpes

655 avenue de l'Europe, Montbonnot

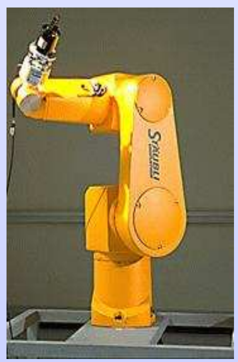
38334 Saint-Ismier Cedex, France

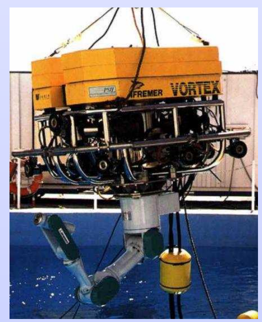
`http://sed.inrialpes.fr/Orccad/`

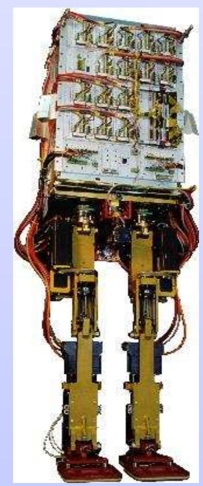
Open Robot Controller Computer Aided Design

ORCCAD : `http://sed.inrialpes.fr/Orccad/`

- A set of concepts, methods and programming tools for robotics
- Handling automatic control and reactivity
- Tools and methods used from design to run-time







General framework

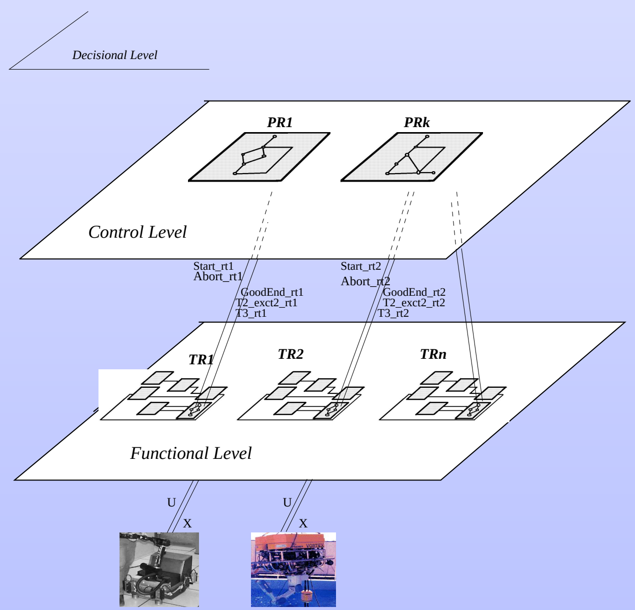
- Complex mechanical systems: many degrees of freedom, non-linearities, coupling
  $\Gamma = M(q)\ddot{q} + N(q, \dot{q})$
- Possible switches in the model (e.g. walking machines, dynamic vision)
- Interaction with the environment, the environment is dynamic and uncertain
- Needs for reliability (e.g. hostile environment, cooperation with humans)
- Hybrid aspects
  - Sampling for sensing and periodic servo-loops
  - Discrete Events for decisionnal processes

Guidelines

- Many robotic actions can be stated efficiently in the framework of Automatic Control theory
  Task-function approach, Sensor-based control. . .
  (*Bottom-up approach*)
- A robotic system has several kinds of users, each of them must be provided with dedicated CAD tools
  (*Different use-cases. . .*)
- Software for robotics must be reliable and reusable
  (*Validation, formal verification, automatic code generation. . .*)
- The overall system performance strongly relies on implementation of real-time programs
  (*Various constraints and co-design*)

Main structures

- Robot-Tasks (RT) model basic robot's actions
  Control law + Logical behaviour
  Control aspects are predominant
- Robot-Procedures (RP): logical and temporal composition of RTs and RPs for incremental design of actions with variable complexity
  Logical aspects are predominant



the Robot-Task: an hybrid object

- Algorithmic aspects:
  Parameterized control law the structure of which is invariant along the RT duration
  ⇒Trees of object classes
  ⇒the leaves are real-time tasks
  ⇒Instantiated via a G.U.I.
- Reactive aspects:
  Specification of the logical behaviour of the RT
  ⇒Encoded with Esterel
  ⇒Coherency with higher levels (RPs)
  ⇒Formal verification

The reactive behaviour encapsulates numerical computations

The RT external view is a Discrete Event System.

Control law design

Task-function : control goal in the task space

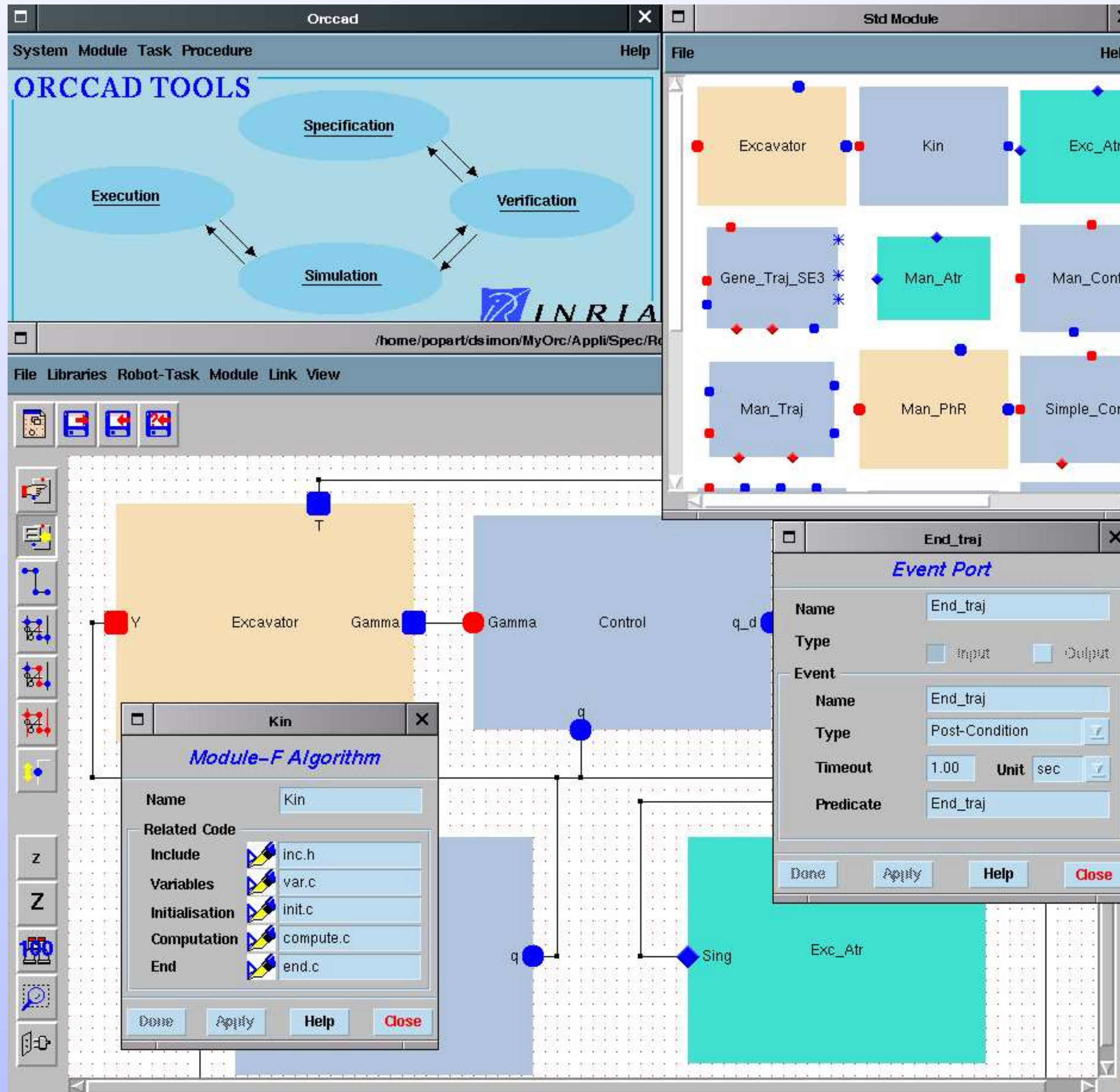
Control law structure:

$$\Gamma = -k\hat{M}\left(\widehat{\frac{\partial e}{\partial q}}\right)^{-1} G\left(\mu De + \widehat{\frac{\partial e}{\partial q}}\dot{q} + \widehat{\frac{\partial e}{\partial t}}\right) + \hat{N} - \hat{M}\left(\widehat{\frac{\partial e}{\partial q}}\right)^{-1} \hat{f}$$

Several control laws can be given according to the choice of models,e.g.

- $\hat{M}$ diagonal and $\hat{N} = 0$: fixed gain decentralized PD
- explicit computation of $\hat{M}$ and $\hat{N}$: computed torque control…

Modular Specifi cation of a Robot-Task

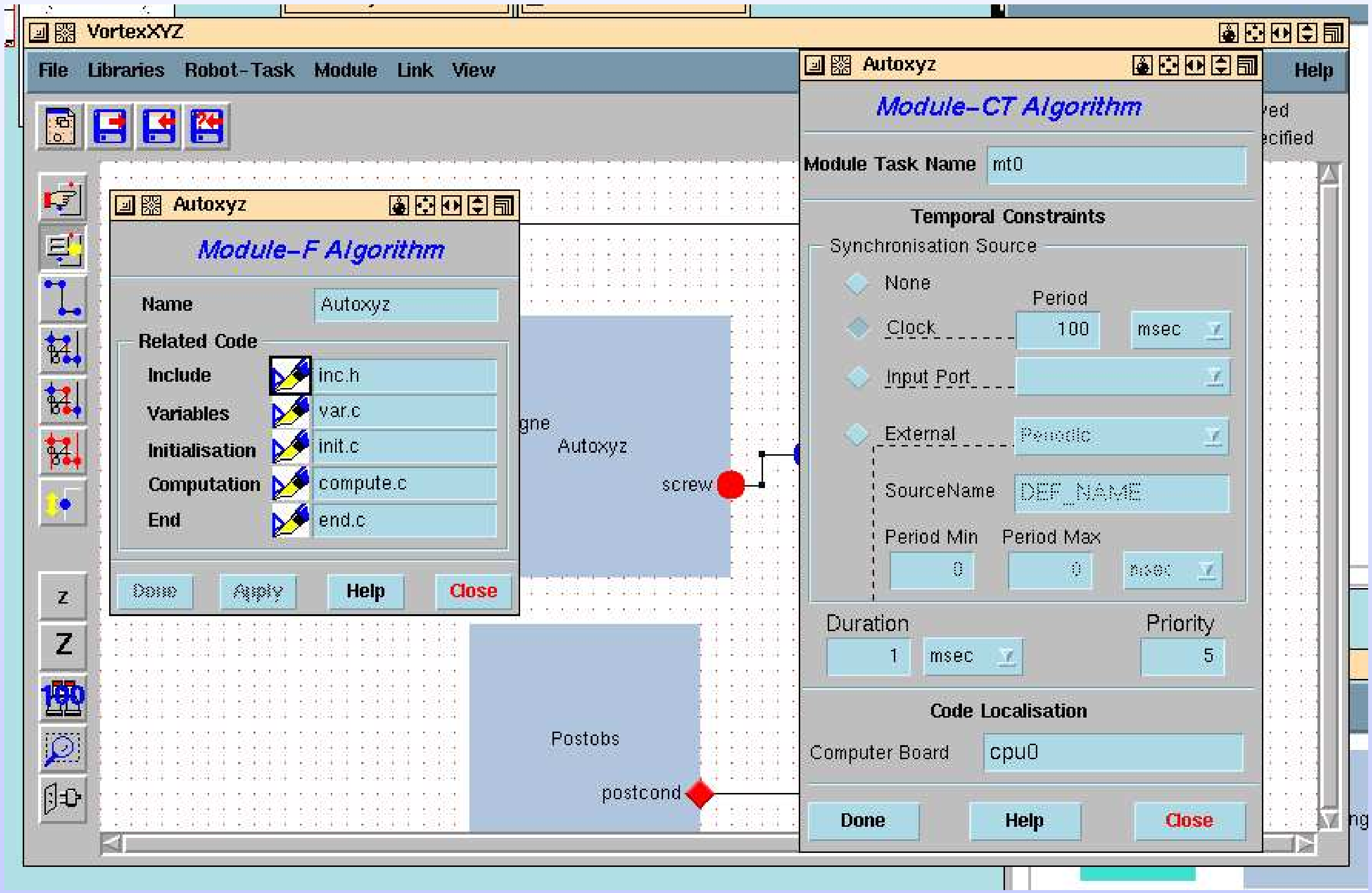


Modules

Several kinds of *Modules*

- class *algorithms*:
  - numerical computation and/or observers
  - temporal attributes : period, duration, synchronizations. . .
- class *physical resource*: interface with the process (drivers)
- class *Robot-task Automaton*: reactive behaviour (e.g. encoded with ESTEREL)

Algorithmic module



- Functional attributes: name and location of C files
- Temporal attributes : Synchro., Priority, CPU ID, WCET

Control and temporal constraints

Goal : achieve control performance w.r.t. available computing power
Theoretic background : few quantitative results, esp. for non-linear systems
Current practice :

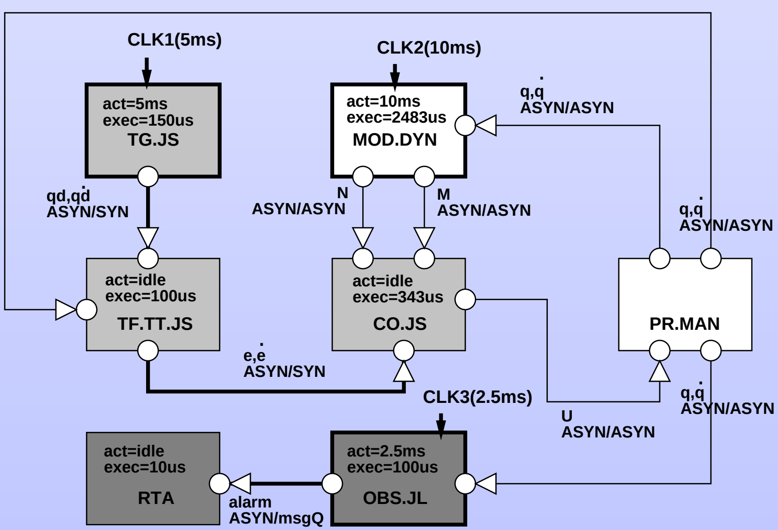
- sampling rate : $0.15 < \omega h < 0.5$ (rule of thumb)
- latency $<<$ period, or change the control algorithm
- jitter ???, very sensitive on identification

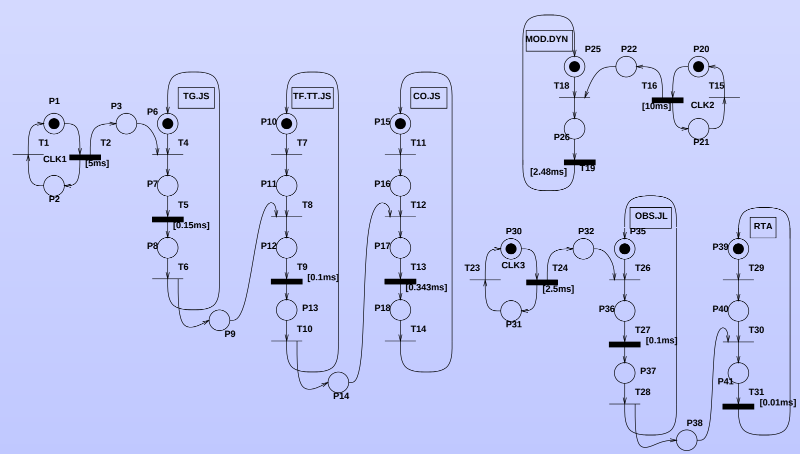
Specify time dependencies through synchronisation
Assign priorities according to the relative urgency of tasks
Synchronised tasks must have the same priority
The best data is the freshest one $\Rightarrow$ single slot links

Take care with shared data and synchronization side-effects

Verifi cation of temporal constraints

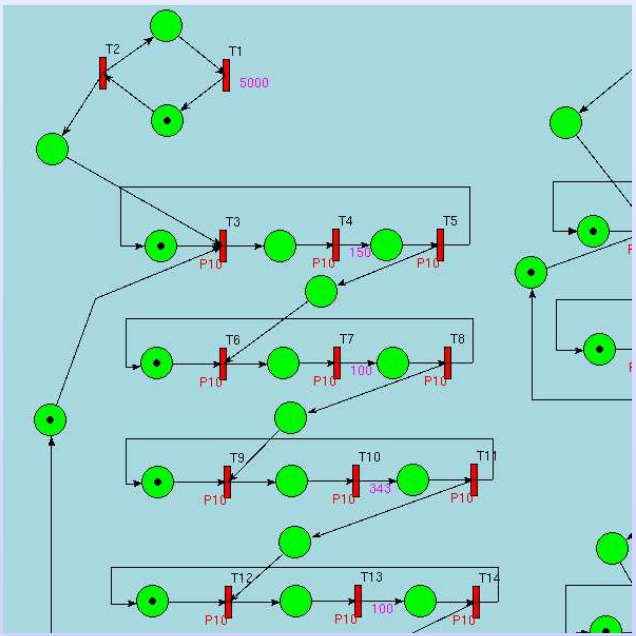
- Partial synchronization to specify precedence constraints
- Timed Petri nets (marked graphs)
- Deadlock analysis by checking the liveness of p-invariants
- Analysis of temporal properties using linear models in the (max,+) algebra (periodicity, stability, respect of deadlines. . .)



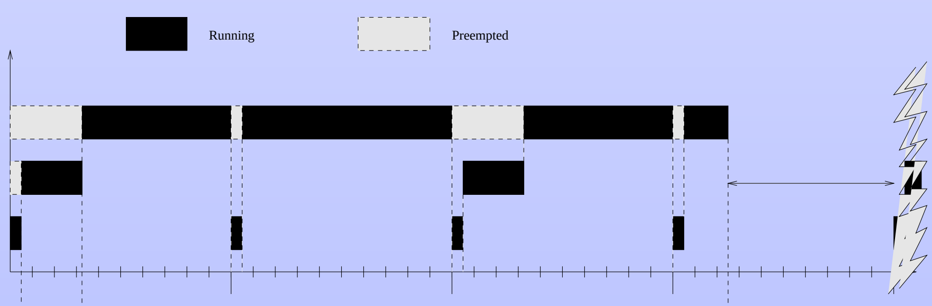


Schedulability analysis

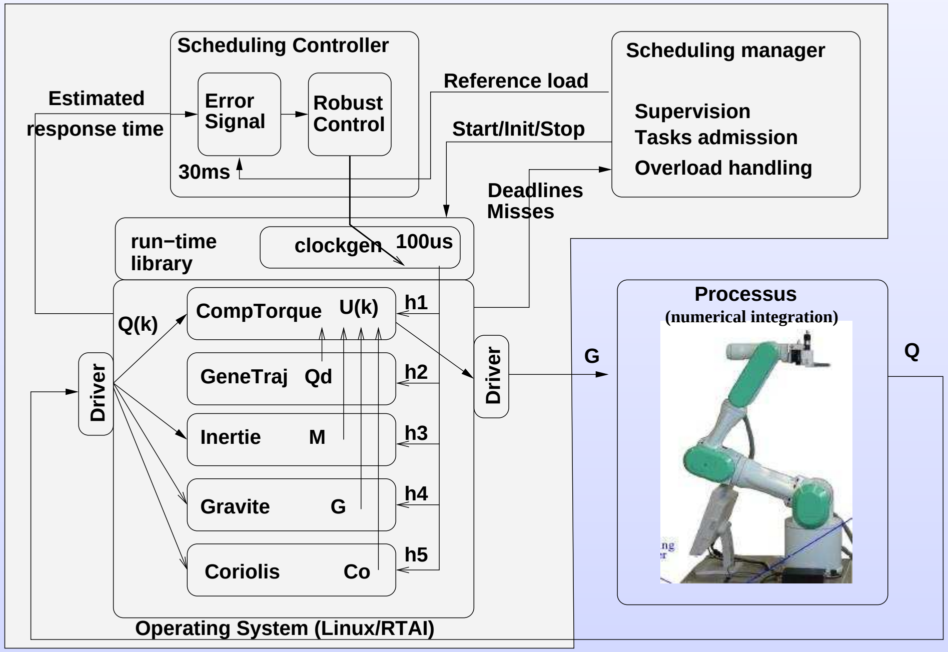
complex synchronisation and scheduling schemes :
ORCCAD GUI => ERS solver in (max,plus) algebra

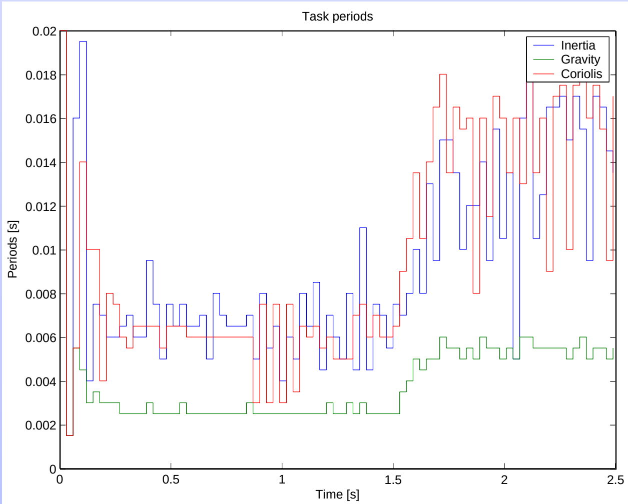



```
Période du système (temps contracté): 8174/1
Période du système (temps réel): 10000/1
DBG: activité autonome (temps contracté):
[][[0,6280[]
DBG: activité préemptée:
1(8106)[0(1894)1(8106)]
Taux d'inactivité: 1894/10000
DBG: activité préemptée (normalisée):
1(8106)[0(1894)1(8106)]
```



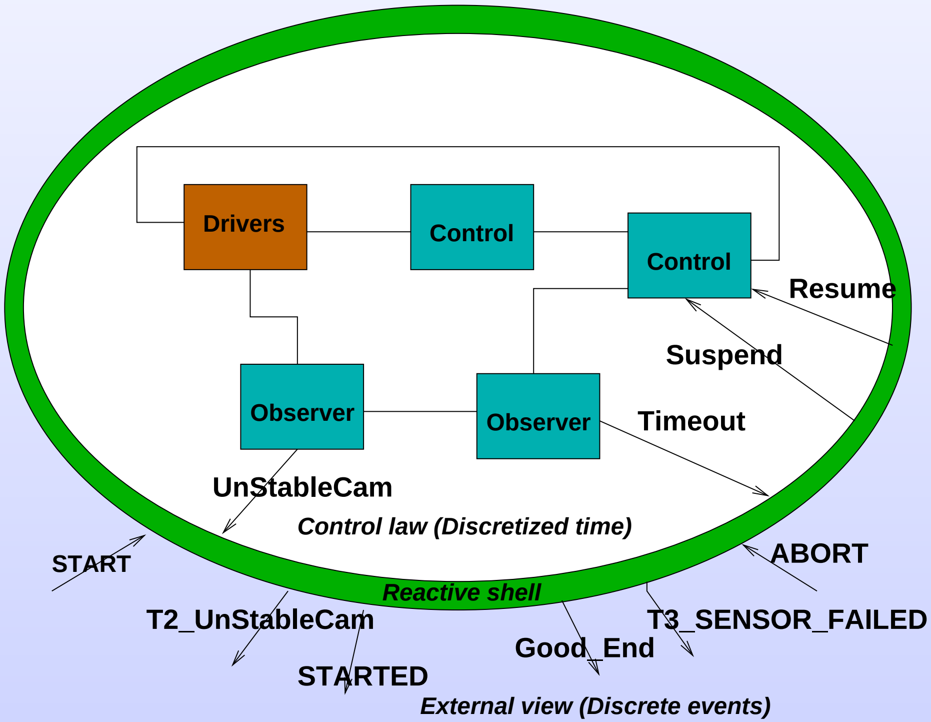
Feedback scheduling





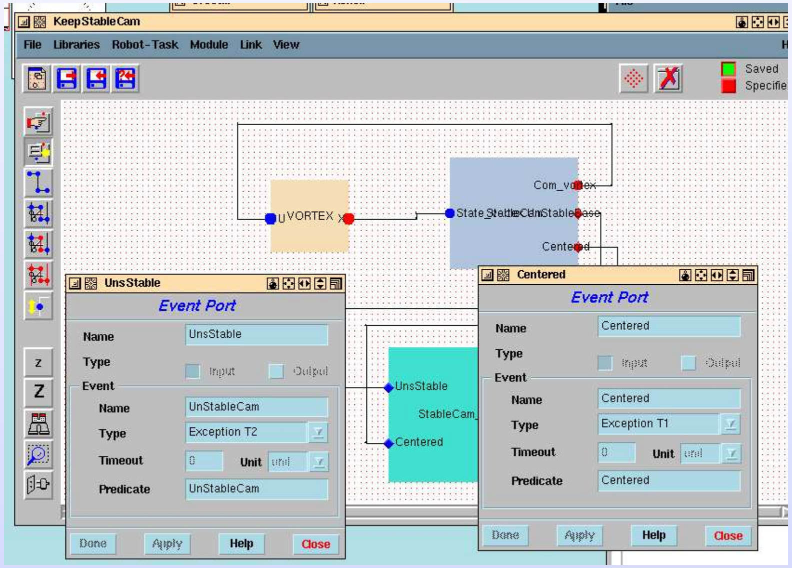
INRIA RHÔNE-ALPES

Events



- **Preconditions**
  - **Synchronization**
  - **Measure (timeout)**
- **Exceptions**
  - **Synchro. signal**
  - **Type 1 (weak)**
  - **Type 2 (strong)**
  - **Type 3 (fatal)**
- **Postconditions**
  - **Synchronization**
  - **Measure (timeout)**

The RTA module




```
trap T2 in
 trap BF in
  [
   await Abort_Local_KeepStableCam;
   emit StartTransite_VORTEX;
   emit KeepStableCamTransite(?KeepStableCam_Start);
   exit Abort;
  ]
 ||
 loop
  await Centered; emit TreatCentered(?KeepStableCam_Start)
 end loop
||
 await UnStableCam; emit T2_UnStableCam; exit T2
 ||
 [[[
halt;
 ]]; exit BF; ]
```

Esterel

- Synchronous language with imperative style
- Modularity using parallelism at specification time
- Powerfull escape mechanisms
- Multiform time
- Deterministic behaviour $\Rightarrow$ formal verification
- Inputs and outputs are signals (pure or valued)
- Communication based on synchronous diffusion
- Compiles into efficient deterministic automata

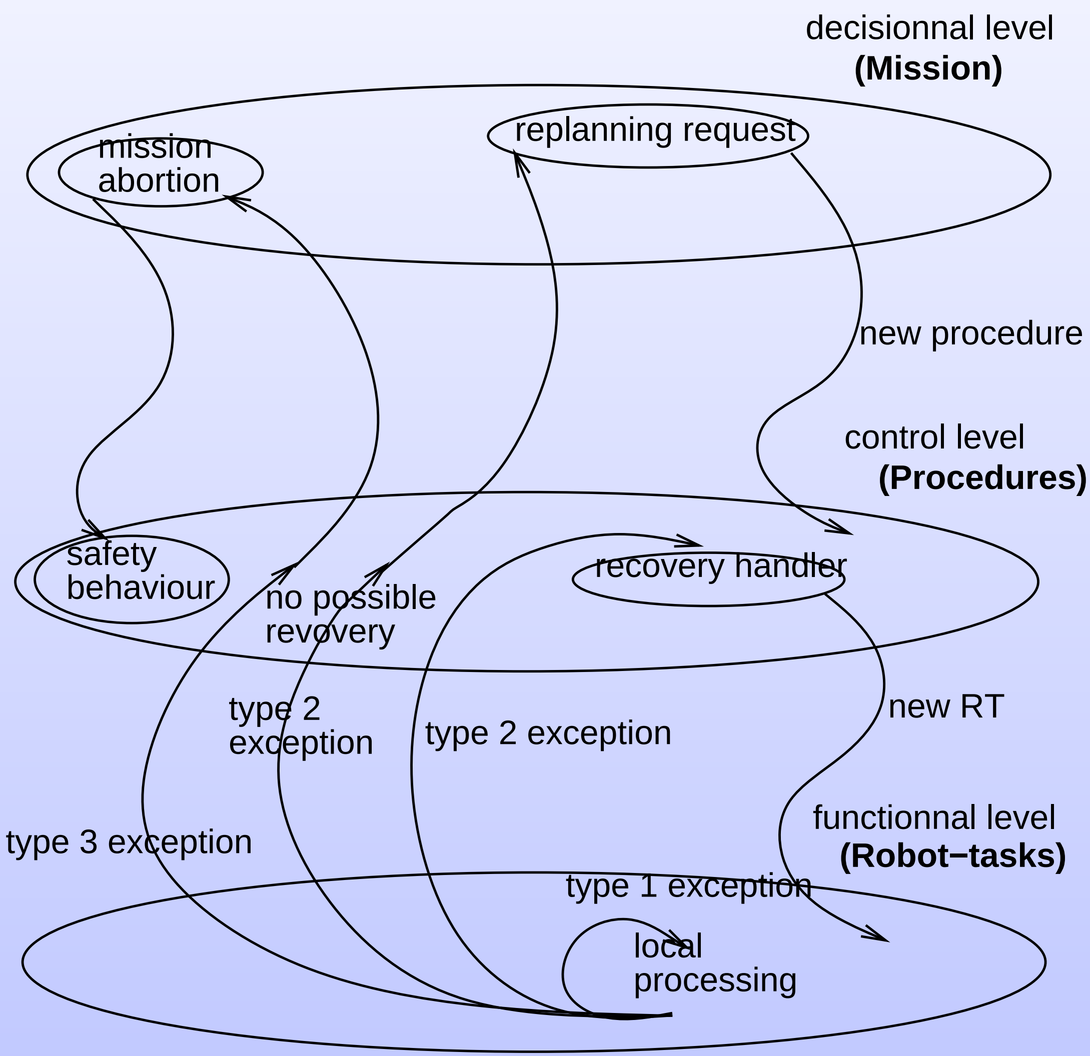
INRIA RHÔNE-ALPES

Main statements

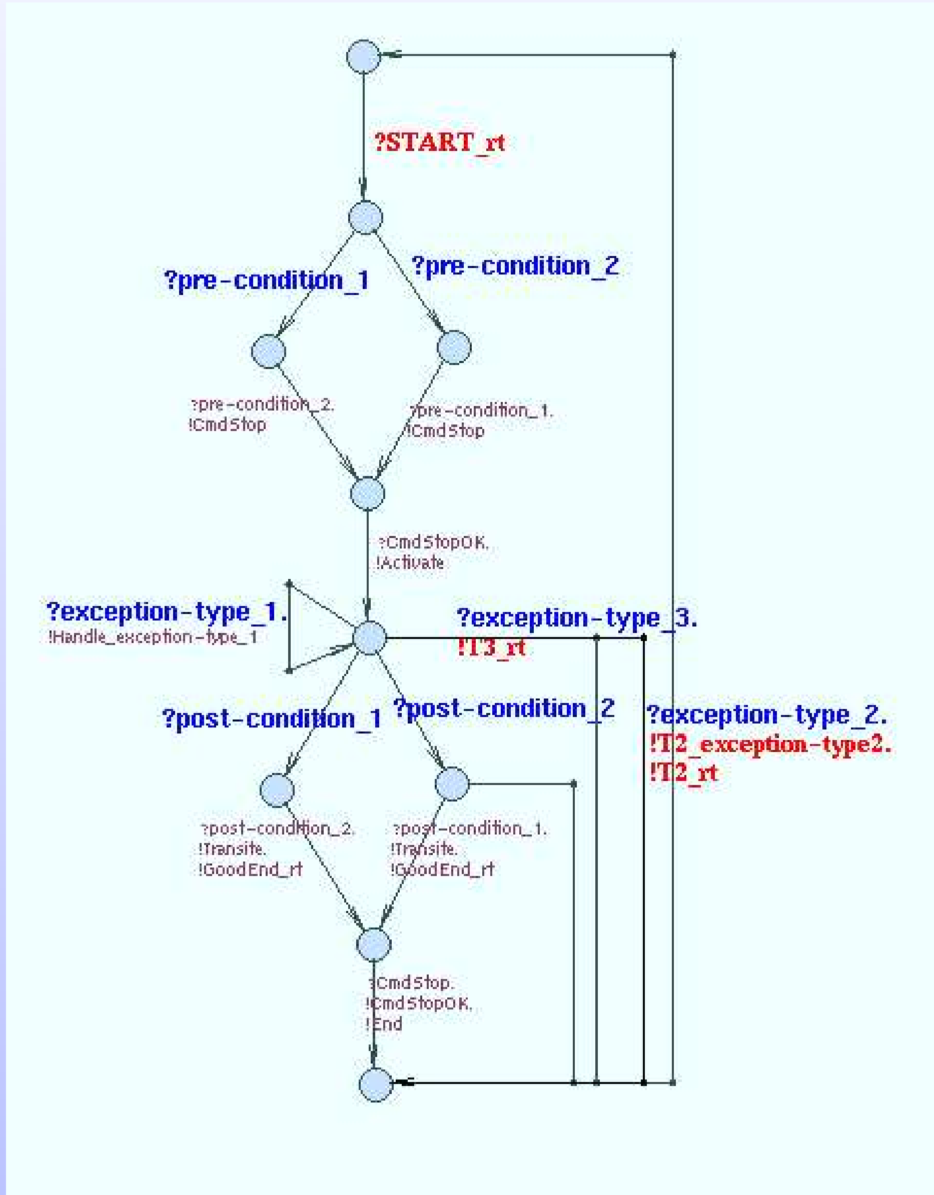
emit S	diffusion of a signal
await S	await the next occurence
p ; q	sequence
p ‖ q	parallel
call P (X1,X2)(e1,e2)	extern procedure call
exec T ()() return R	extern task launch
loop p end	infinite loop
loop p each S end	temporal loop
present S then p else q end	branching
abort p when S	strong preemption
weak abort p when S	weak preemption

POP ART

Exception processing



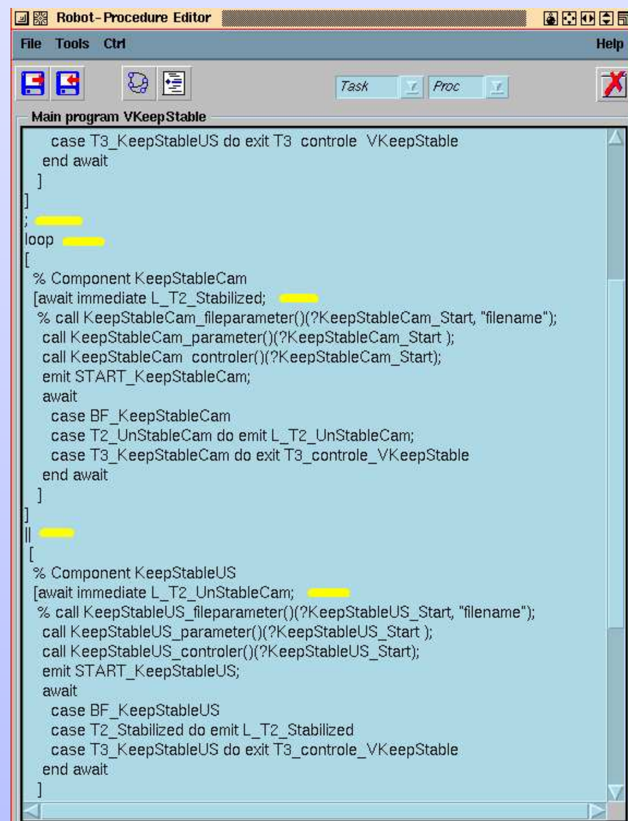
A Robot-Task Controller



Robot-Procedure specifi cation

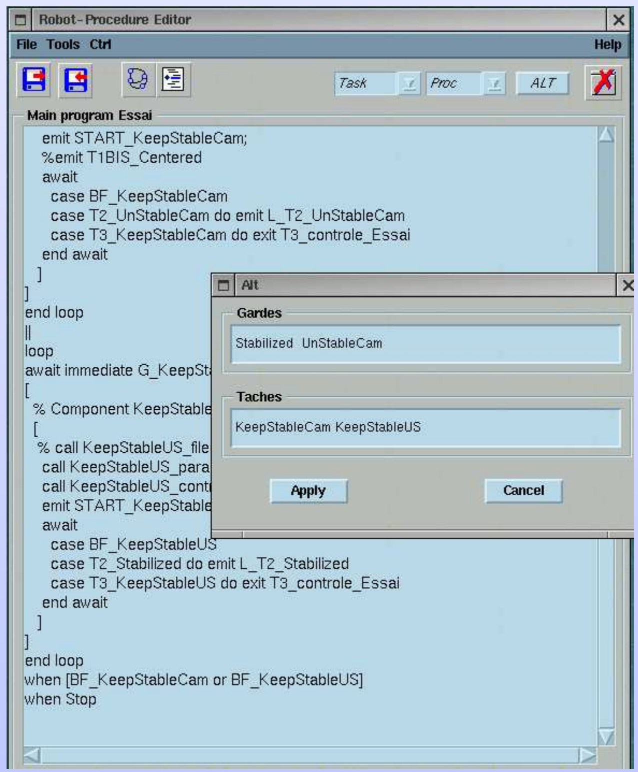
- Incremental design of complex actions (nominal and degraded modes)
- Exception handling
- Semi-automatic code generation
- ESTEREL : sequence, parallel, loops, watchdogs, traps. . .

  We don't have a FSM in mind at design time



Specifi cation of a guarded alternance

- actions are guarded by T2 emitted by other actions
- actions are mutually exclusive
- no dead-lock nor endless loop



Maestro : a domain specifi c language

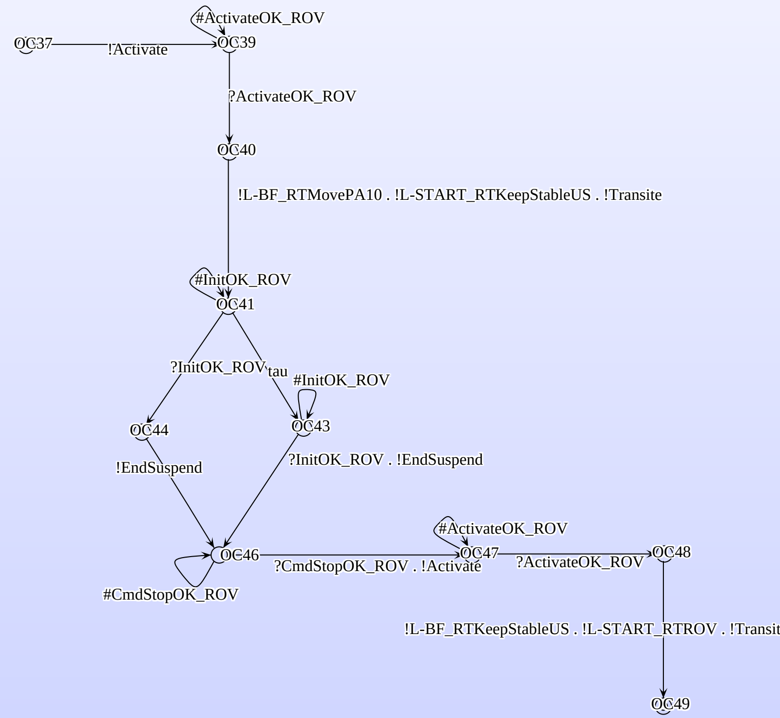
```
do
  SEQ(do
        KeepStableUS
    until Stabilized
    ;
    loop
      PAR(when T2_Stabilized  when T2_UnsStableCam
          do                  do
            KeepStableCam        KeepStableUS
          until UnStableCam    until Stabilized)
    end loop
     )
until Stop
```

a graphical version would be better???

INRIA RHÔNE-ALPES

Switching RTs

Robot Task 1	Robot Task 2	Switching signals from/to automaton	Comments
		post−condition or type 2 exception	Decision for switching to RT2
		Transite	
		Resume	Resume real−time tasks (MTs)
		Resume_ok	
		Init	Task initialization Checking for pre−conditions
		Init_ok	
		Control_stop	Stop of control application
		Control_stop_ok	
		Control_start	Start of control application
		Suspend	
		Suspend_ok	



Complex logic + real-time scheduling (overlap during transite)

POP ART

Formal verifi cation : requirements

Once a mission has been defined, check that:

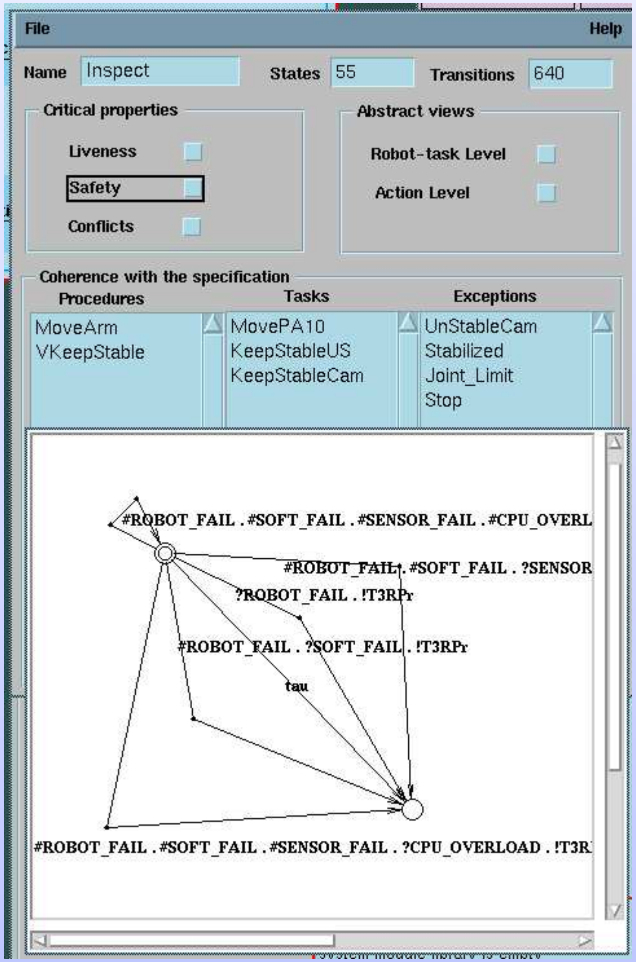
- its specification is correct (corresponds to the desired goal)
- its programming conforms to specification
- real-time implementation does not disturb its behaviour

- Safety properties (fatal exceptions are always correctly handled)
- Liveliness properties (the goal is always reached in nominal executions)
- Conflicts detection (mutual exclusion)
- Conformity with the requirements
- Help to specification (abstract views)

Safety properties

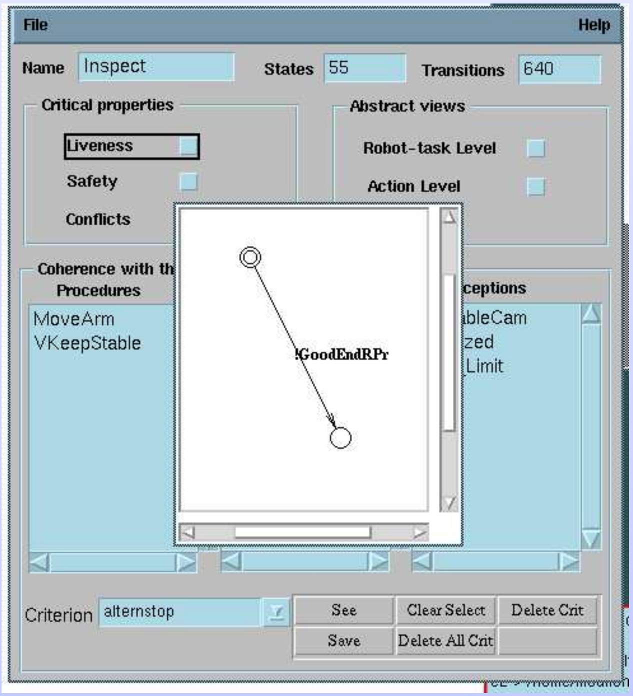
fatal exceptions are always correctly handled

abstract action: “$Error = /Water\_Leak?\ and\ not\ /Accent!$”



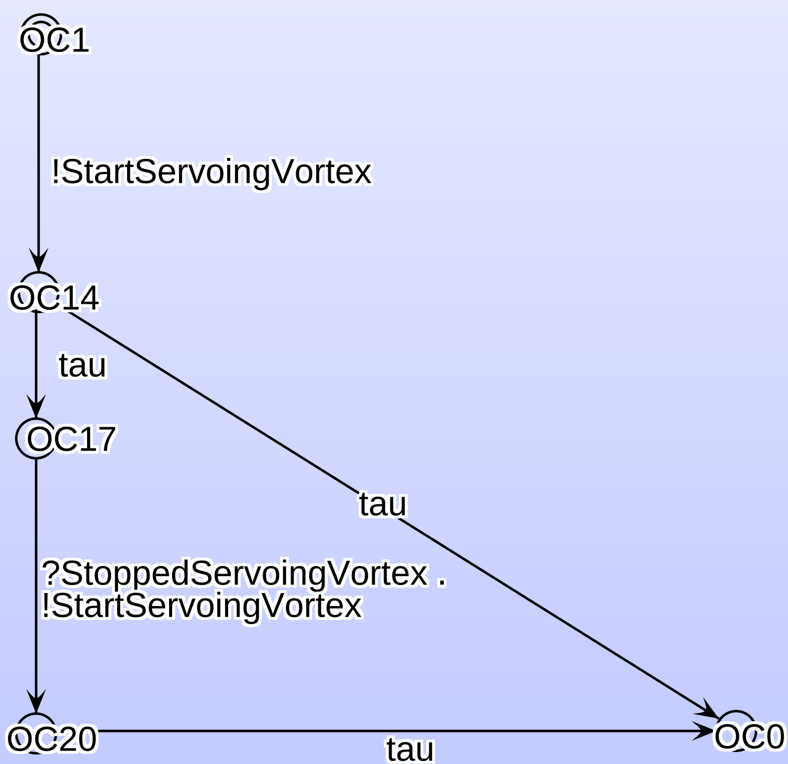
Liveliness properties

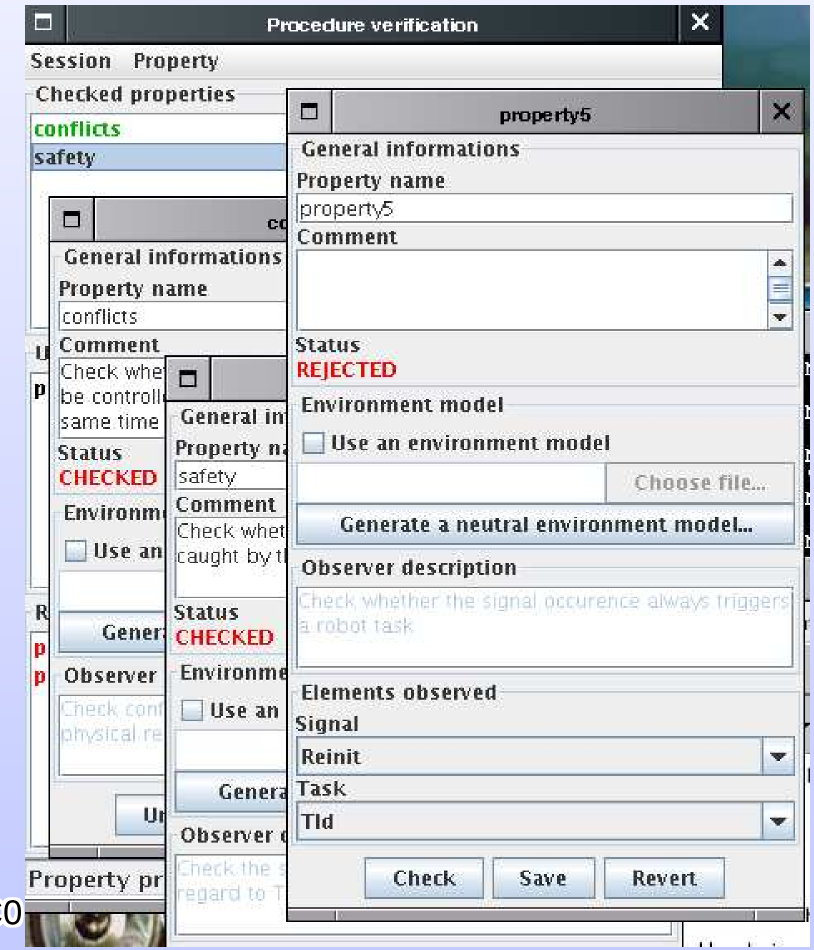
the goal is always reached in nominal executions
e.g. there are no endless loops. . .



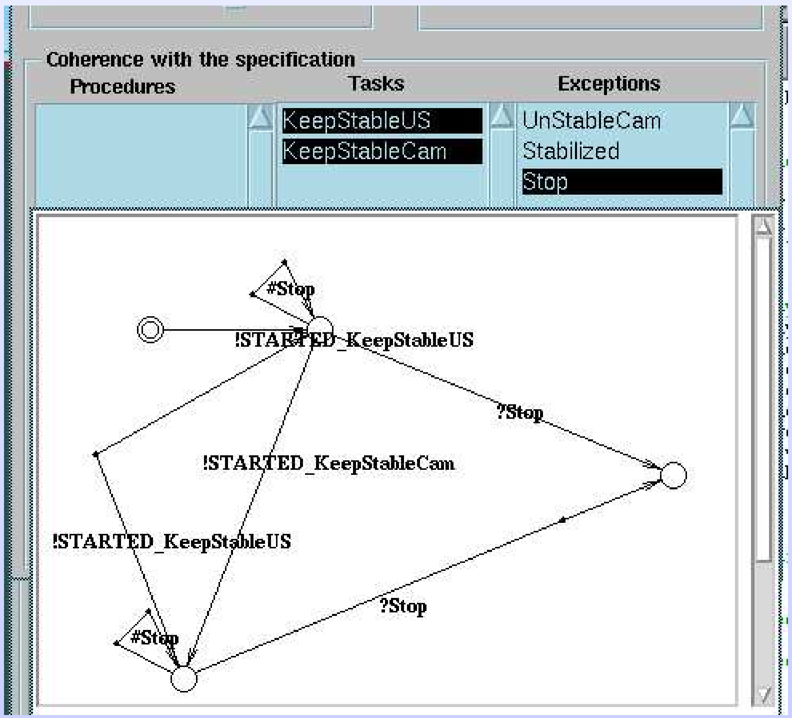
Conflicts detection

two control laws must not compete to control the same robot



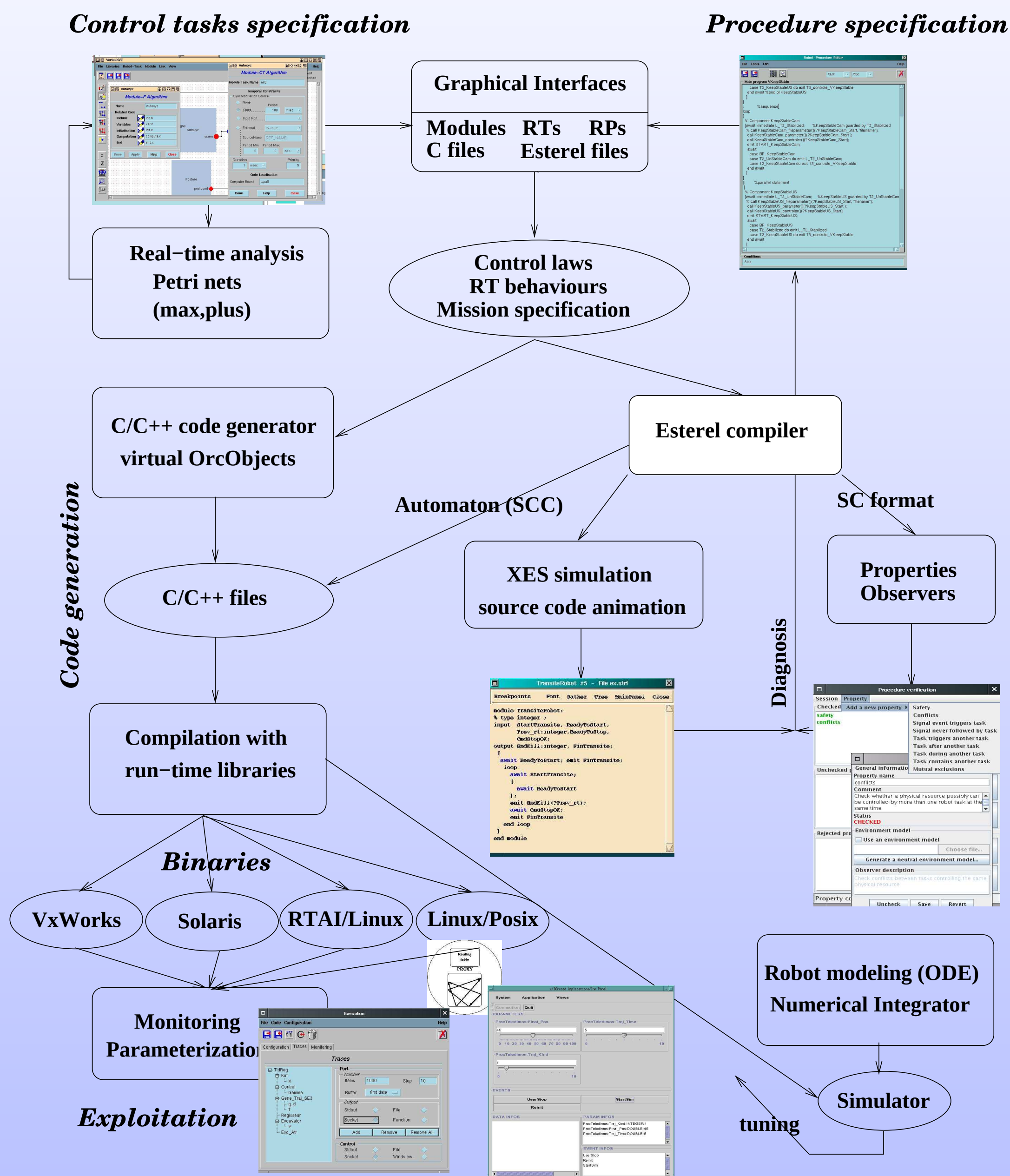


Conformity with the requirements

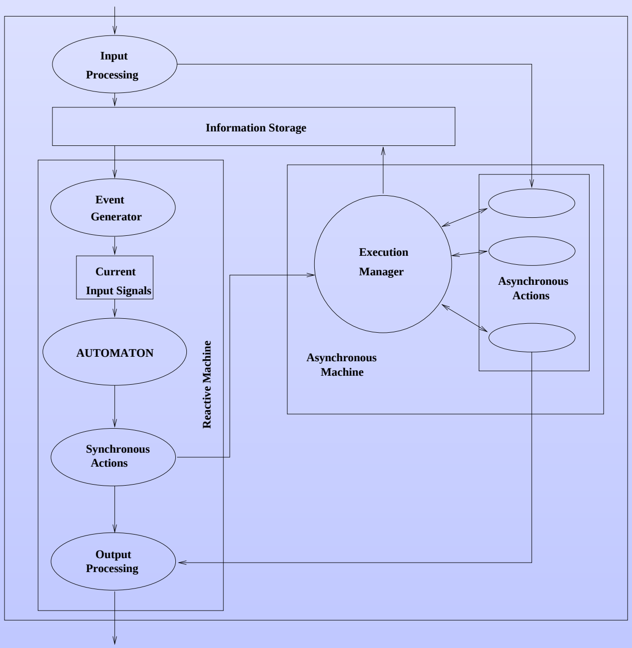


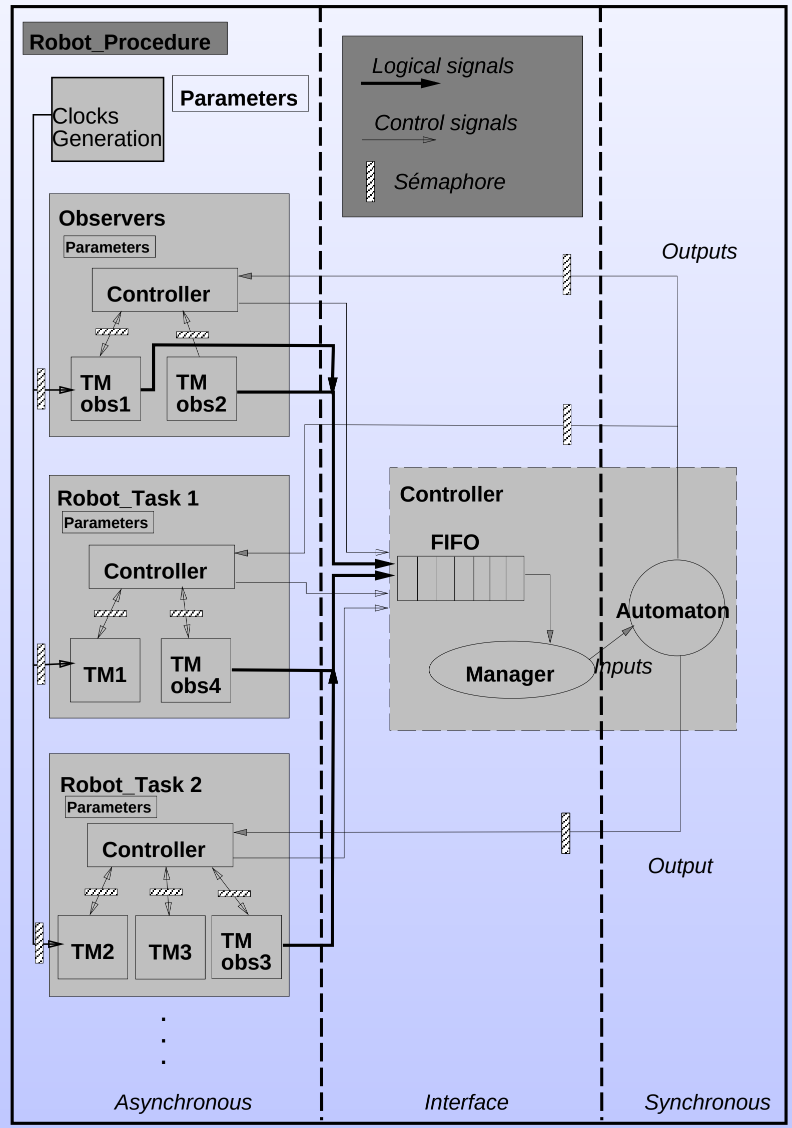

```
Start KeepStable;
weak abort
        Start KeepStableUS;
        [await T2_Stabilize
        Start KeepStableCam
        await T2_Unstable
        ||
        await T2_Unstable;
        Start KeepStableUS
        await T2_Stabilized
when Stop
```

Summary of Orccad tools



Real-time implementation

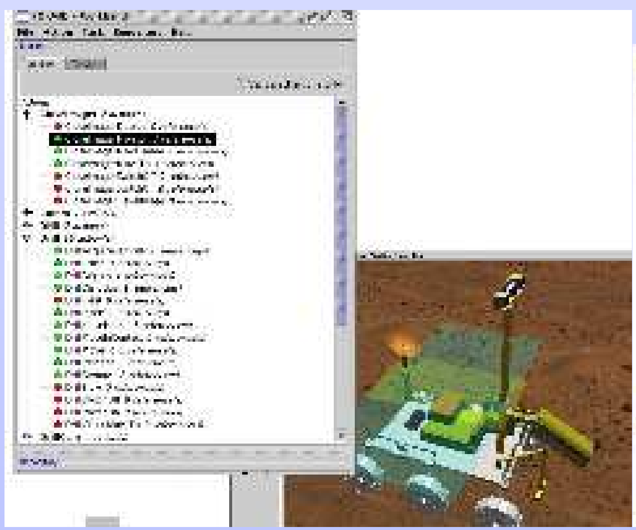




only uses the basic features of the RTOS

Using Orccad tools

- BIP2000 designers : control and/or mechanical engineering
- flexible and feedback scheduling
- underwater arm/vehicle high-level coordination
- specification/simulation of missions for a Mars rover
- control of vision-based tasks (Vimanco)
- teleoperation interface with force-feedback (Teleman)



Current release : Orccad V3.1

- Edition of modules : algorithmic, automaton and PhR
- Edition of Robot-Tasks : single loop, multi-rate
- Procedures : Semi-automatic generation of Esterel, user-level language (MaesTro, no longer maintained)
- GUI built on IlogViews
- Generation of C++ classes or C structures
- targets : VxWorks, Solaris, Linux, RTAI
- Verification : generic properties, Atg, observers
- Monitoring interface
- Maintained by Inria-RA staff

Current improvements (and dreams)

- Open-Source re-design (Kernel + plugins)
- Generic simulation templates
- Code distribution (including fault tolerance)
- Teleoperation (e.g. distributed vision based robot control)
- Flexible scheduling based on feed-back scheduling
- Graphical specification of procedures (Grafcet or StateCharts like)
- More friendly verification tools
- Discrete events controller synthesis
- ...

Model Driven Architecture emerging standards and tools