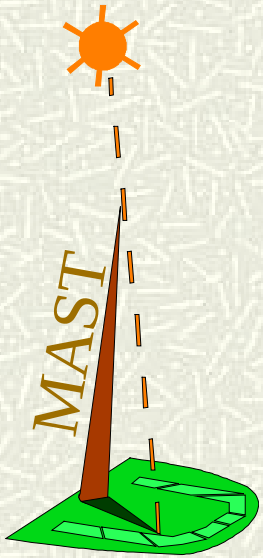
Modeling and Schedulability Analysis of Hard Real-Time Distributed Systems based on Ada Components*



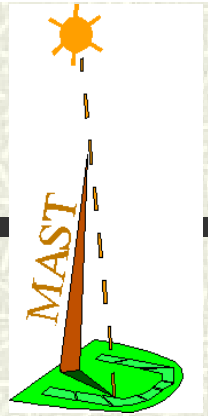
J.L. Medina, J.J. Gutiérrez, J.M. Drake, and M.González Harbour

{medinajl,gutierjj,drakej,mgh}@unican.es

University of Cantabria

Santander (Spain)

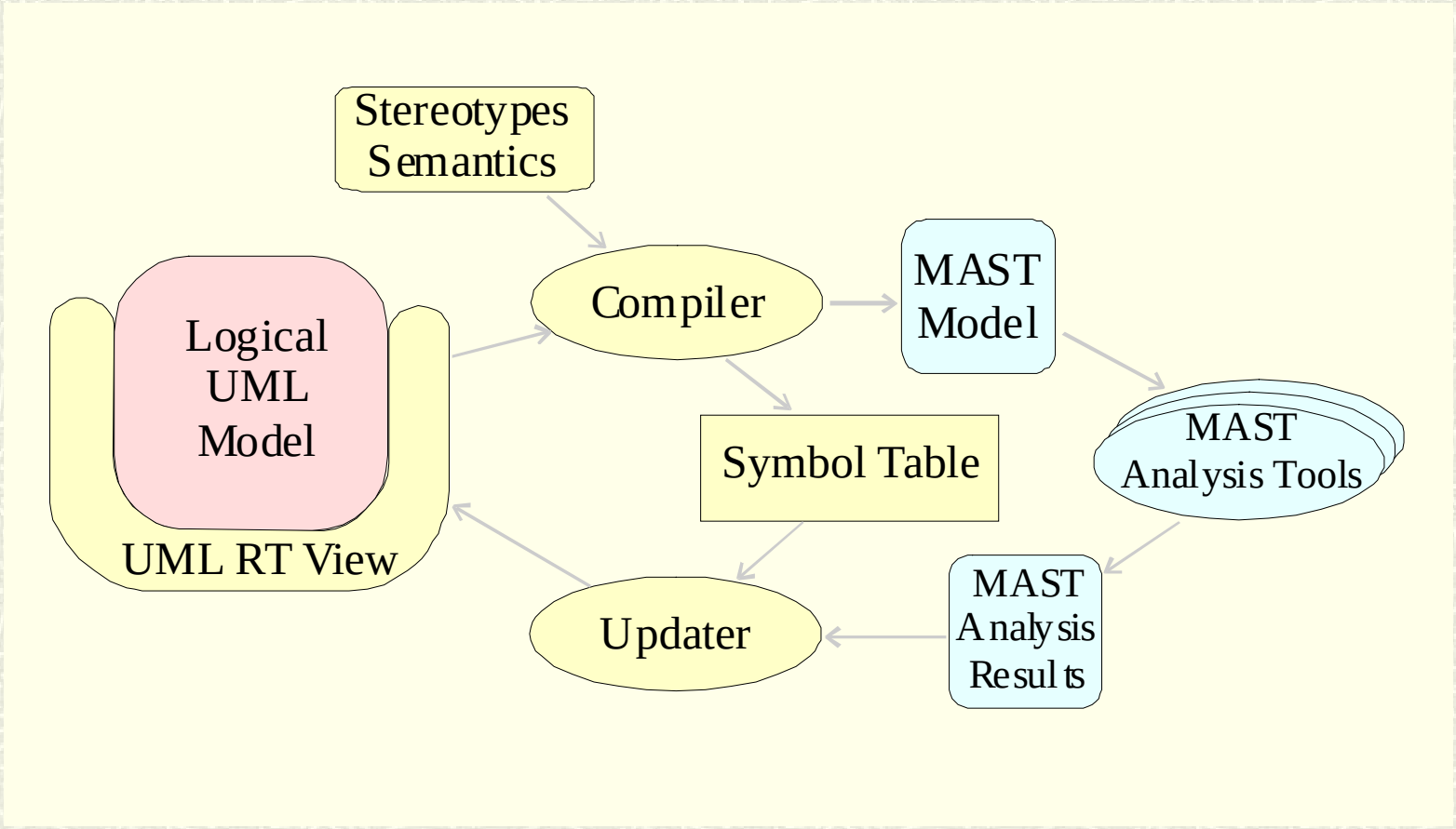
[*] Funded by CICYT (TIC99-1043-C03-03 and 1FD 1997-1799)

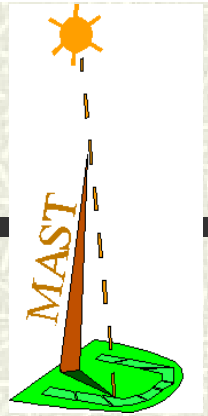


Objectives

- Real-time modeling and analysis of applications written in Ada 95 and using Annexes D and E.
- Goals of this methodology:
  - Based on independent models of the Platform, the Application software components, and the Real-time situations.
  - The semantics of the modeling components include fine details of the Ada structures.
  - Reusable models of the logical Ada components.
  - Automatic modeling of local and remote access to distributed services.
  - Formulated with UML: may be supported by any standard CASE tool.

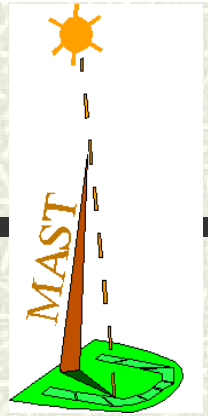
Modeling and analysis and process



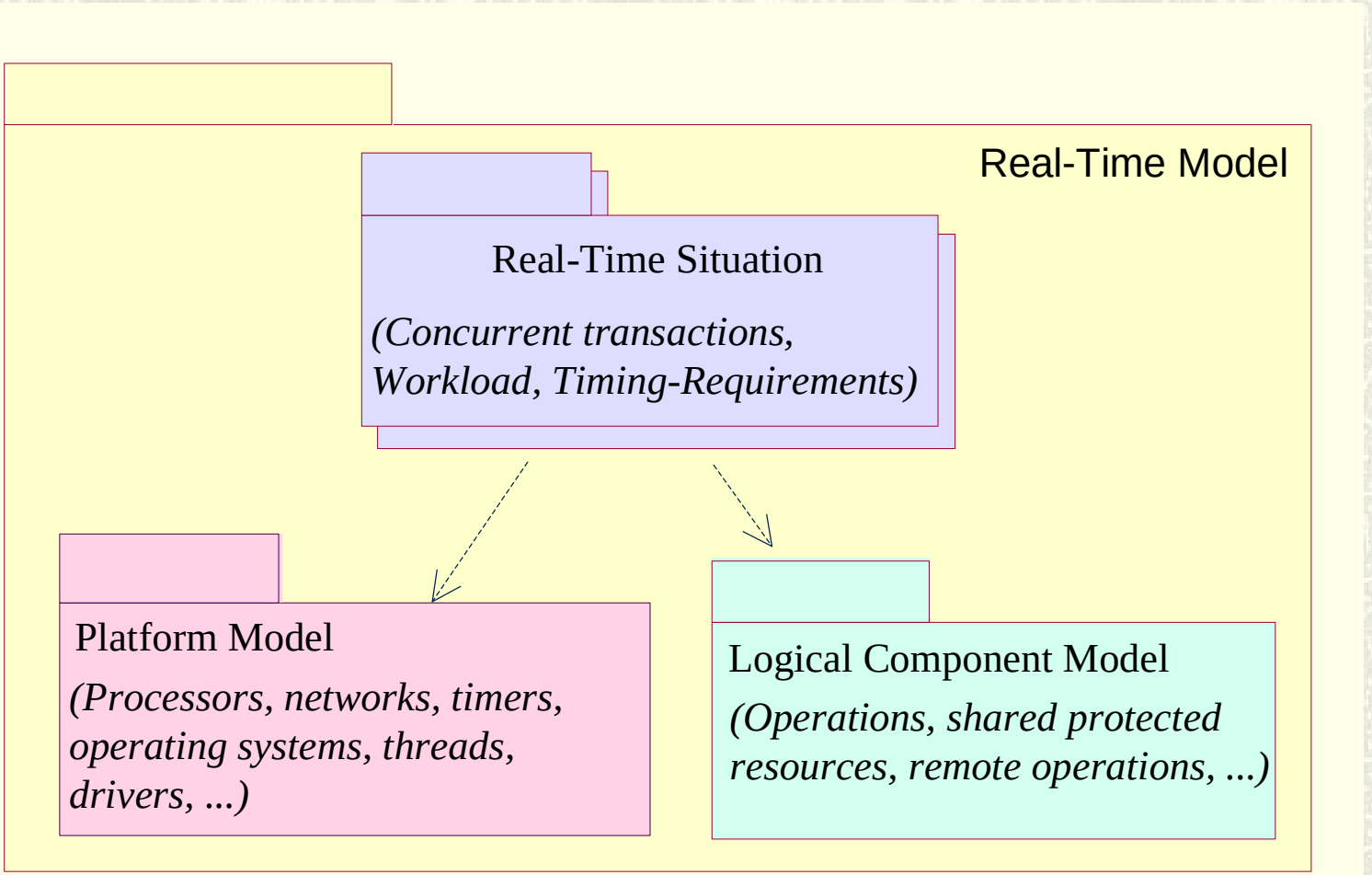


Analysis and design tools

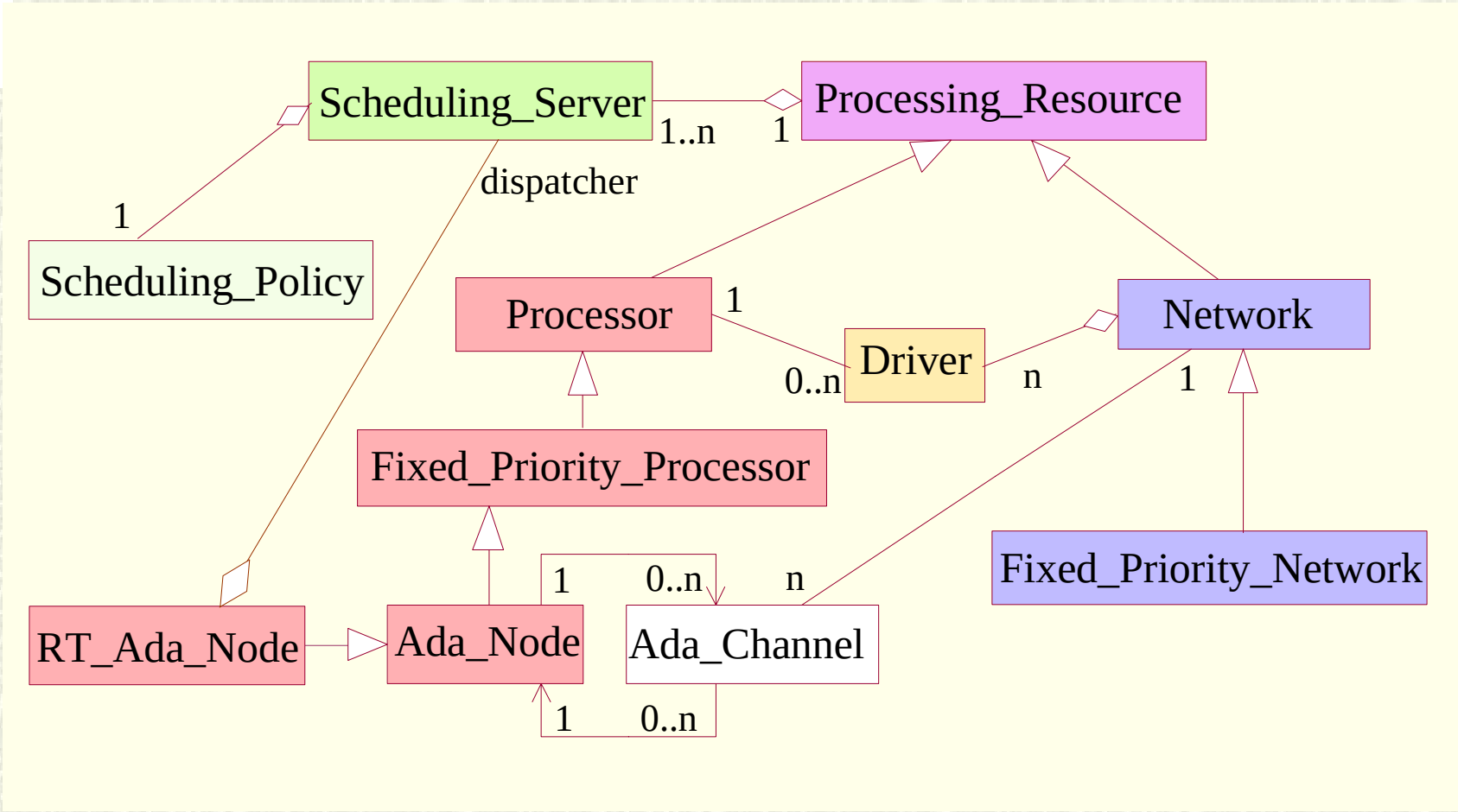
- Available tools:
  - Holistic analysis
  - Offset-based analysis
  - Varying priorities analysis
  - Multi-processor priority assignment
    - Linear HOPA
    - Linear simulated annealing priority assignment
- Tools under development:
  - Multiple event analysis
  - Multiple event priority assignment
  - Mono-processor and distributed simulation

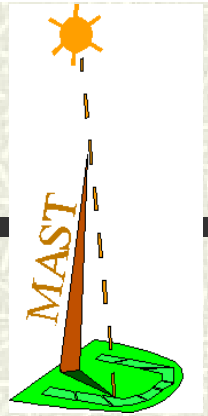


Sections of real-time models

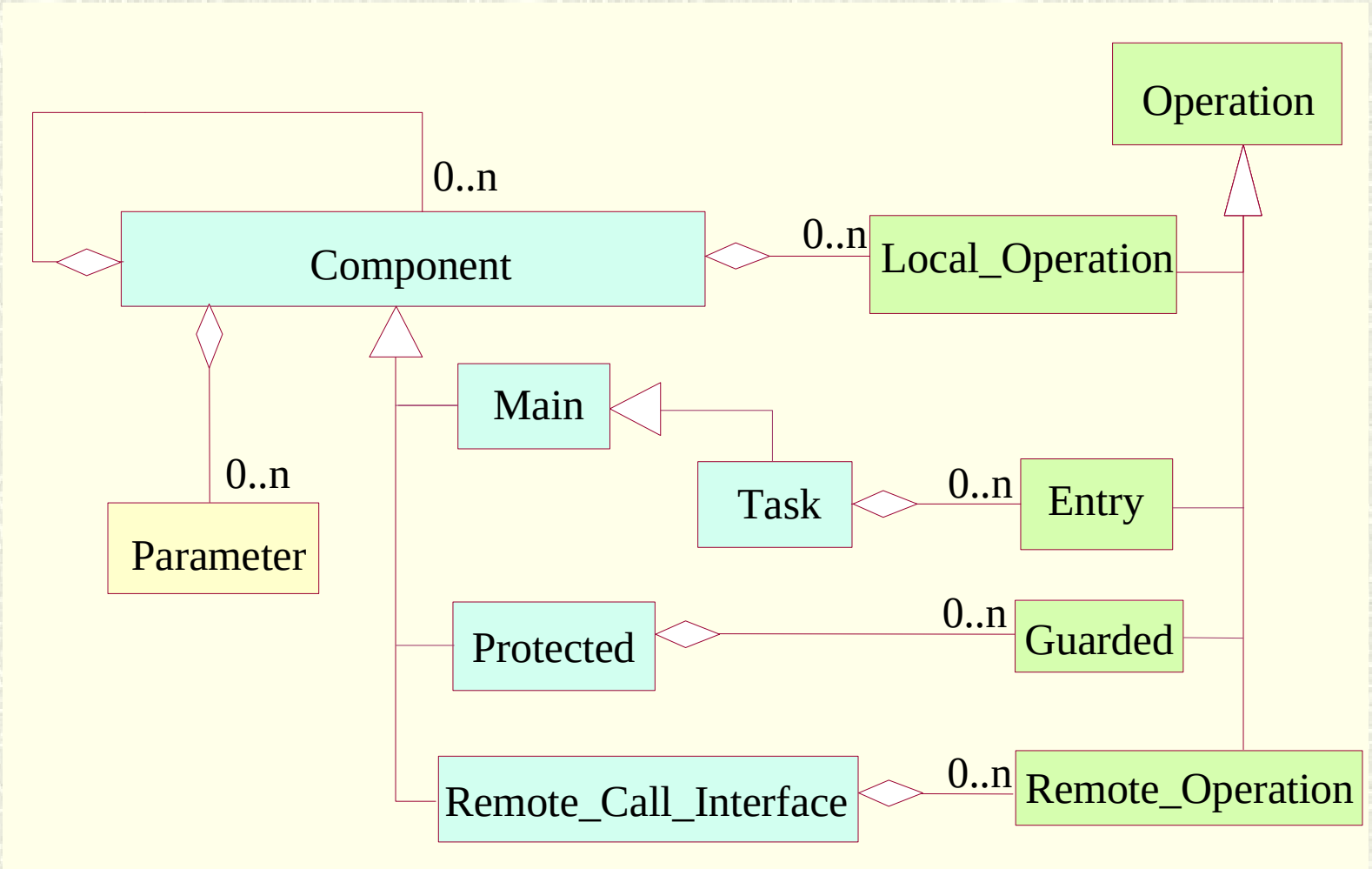


Platform model

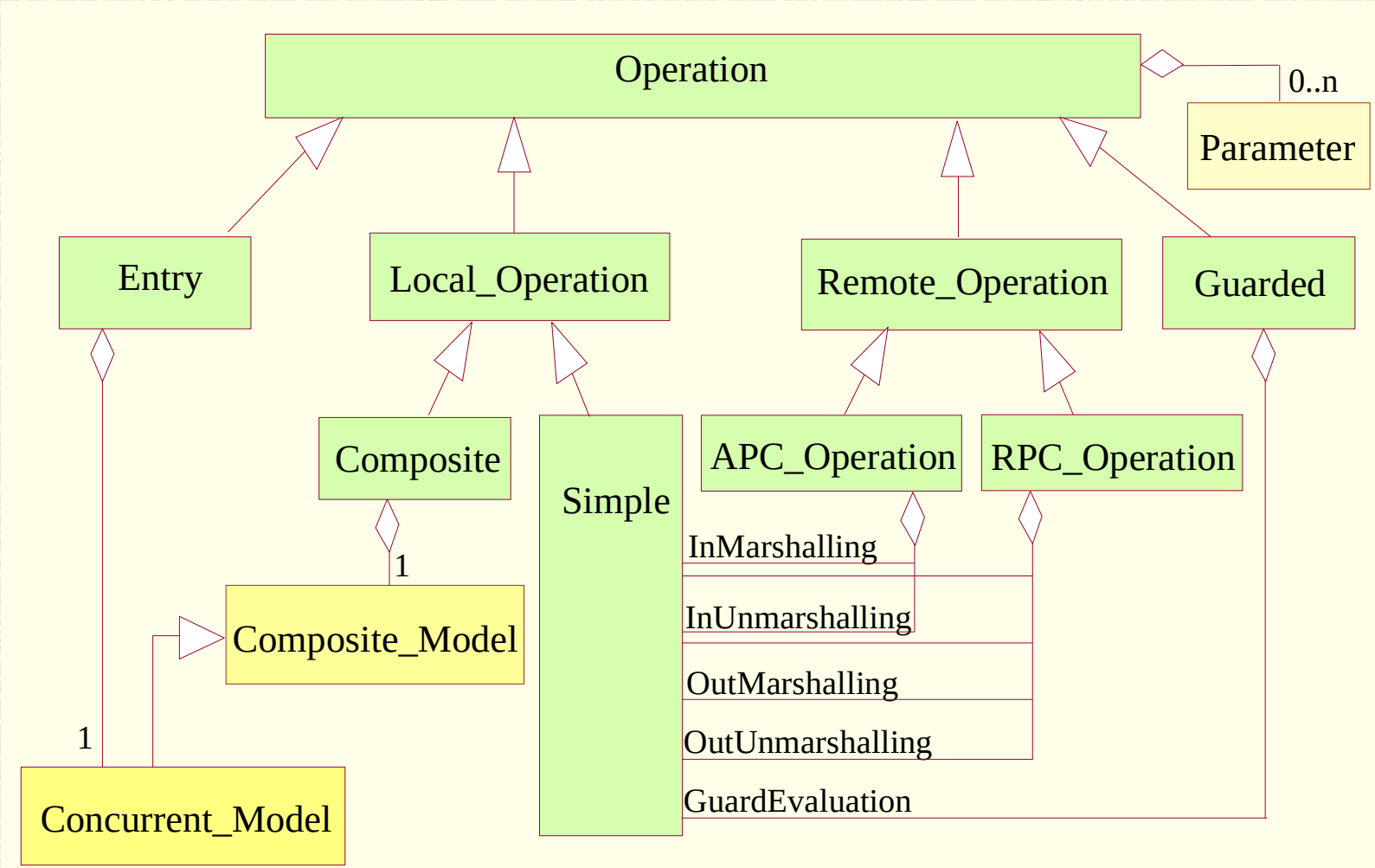




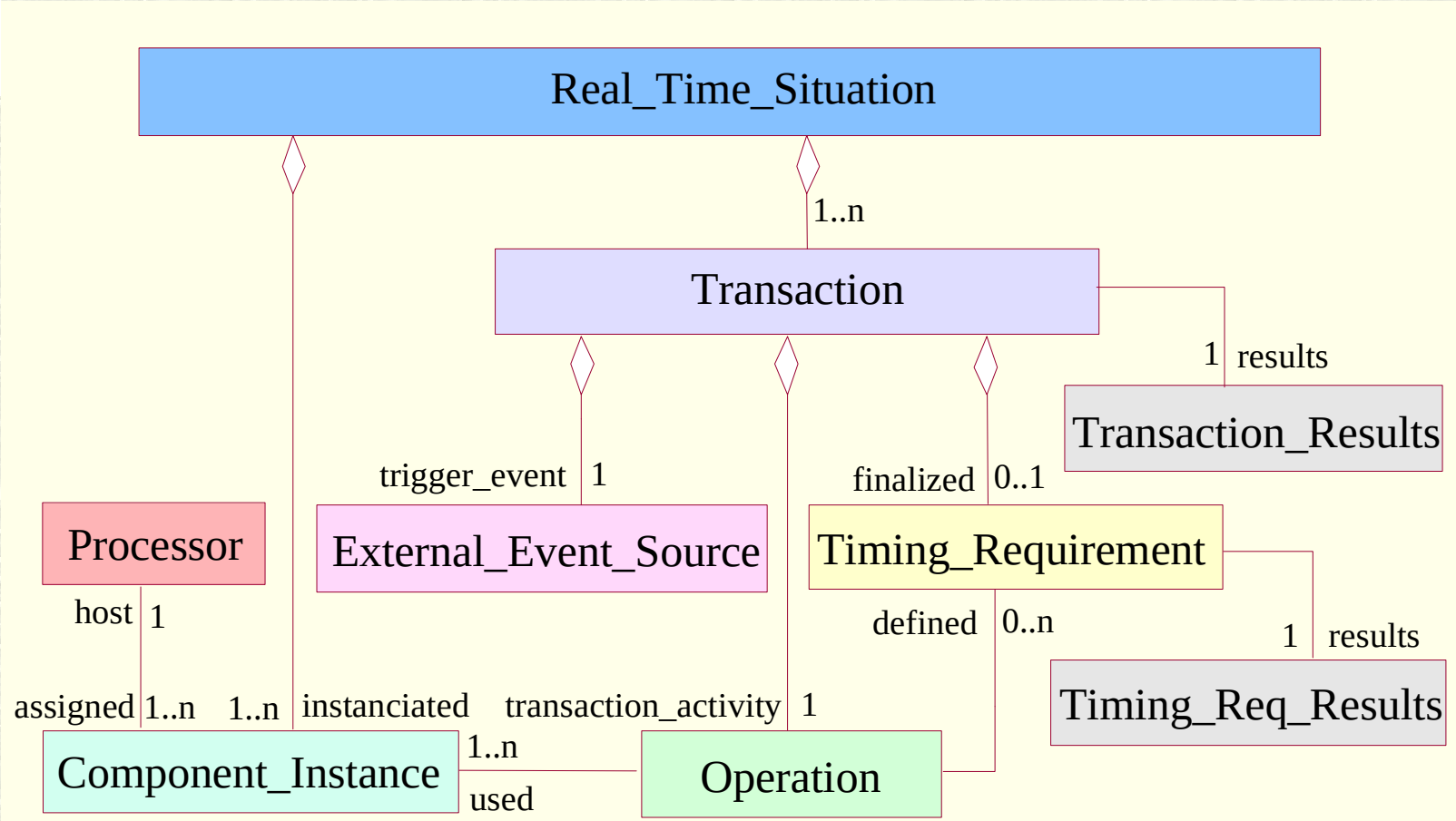
Classes for modeling the logical Ada structures

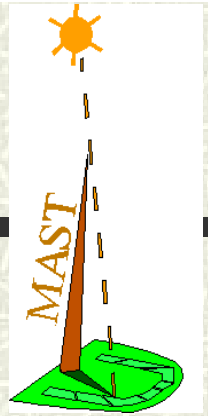


Classes for modeling procedures and functions



Classes for modeling the real-time situations



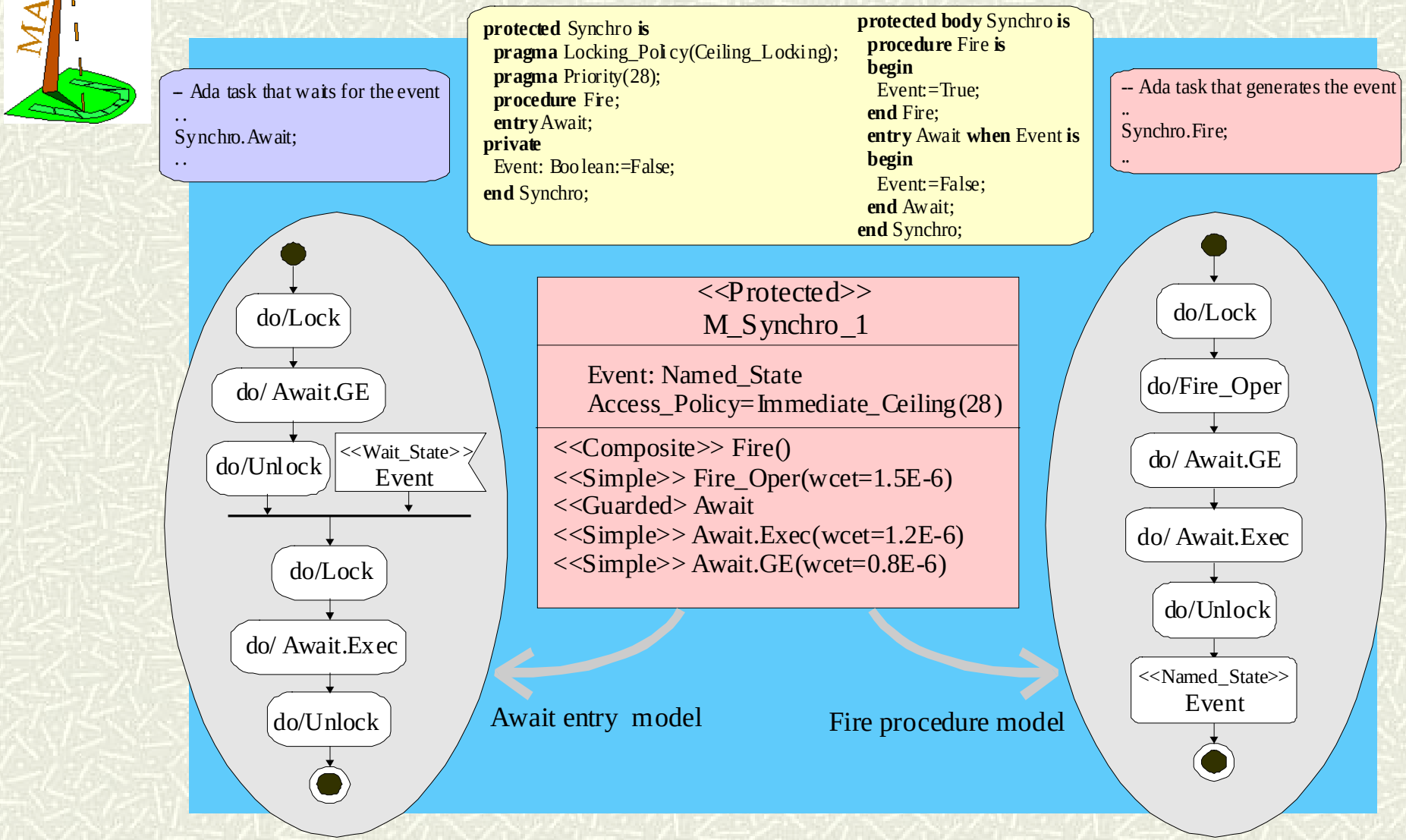


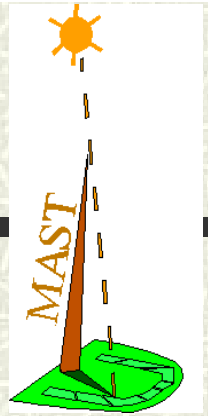
Suitability for modeling Ada structures

- Modeling the structures:
  - The models of the Ada components (packages, tagged types, tasks, protected objects, etc.) are reusable and application-independent.
  - The model preserves the same structure (dependency, visibility, naming conventions, scope) of the Ada application.

- The model includes the timing behavior details of:
  - Synchronization primitives (protected object access, task redezvous, interrupt service, etc.)
  - Ada tasks.
  - APCs and RPCs.

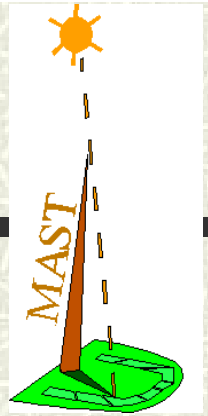
Ada constructs: model of a synchronization artifact.





Ada constructs: synchronization model limitation

- A general protected object can not be modeled with this approach, since:
  - Guard conditions are arbitrary
  - Requeues can create arbitrary dependencies
- We can model the most frequent synchronization patterns in real-time applications, like:
  - One task signals another one
  - Broadcast: one task signals many
  - Barrier: many tasks activate one
  - ....



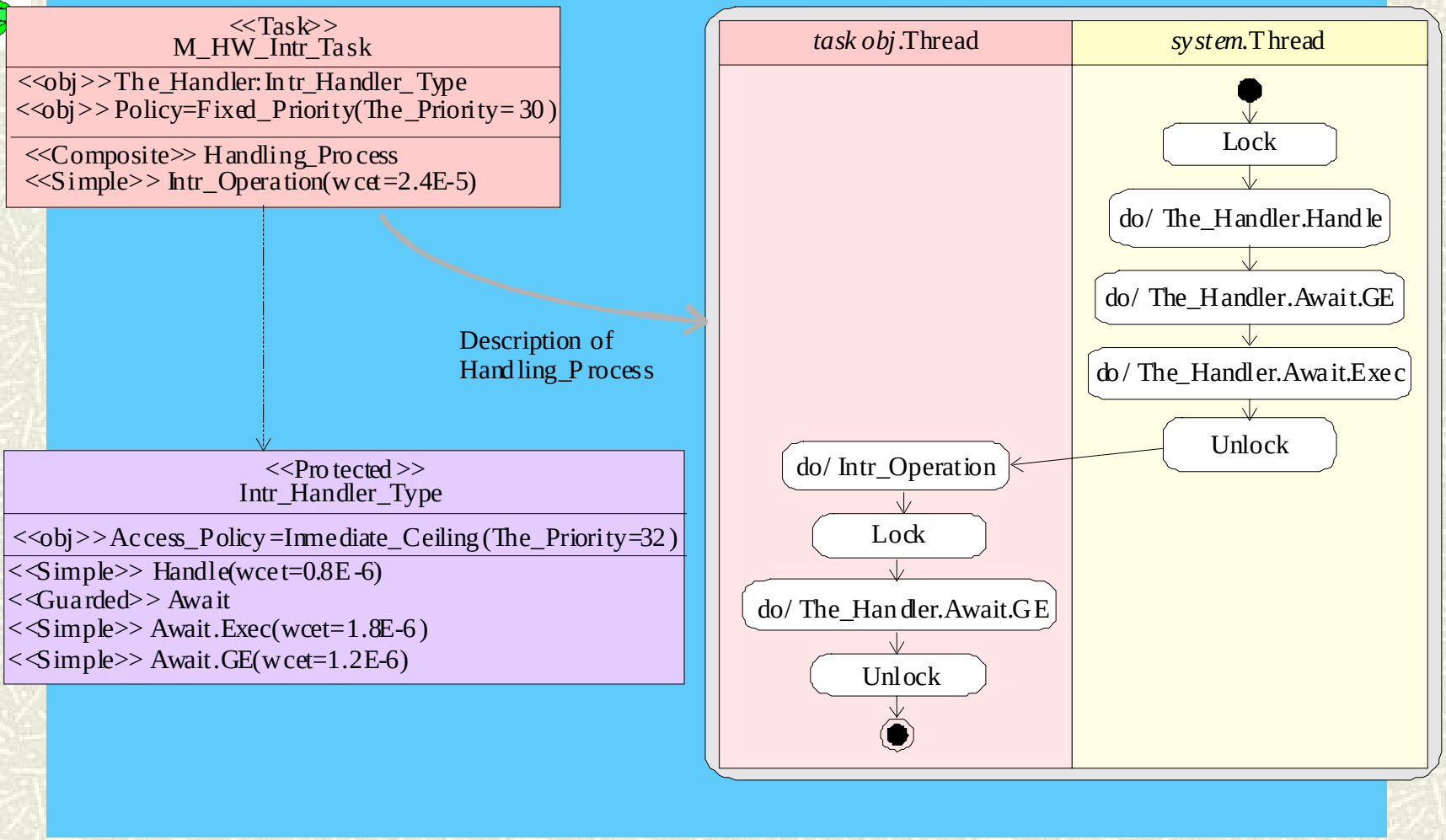
Ada constructs: hardware interrupt ada code

```
task type HW_Intr_Task;

task body HW_Intr_Task is
   The_Handler : Intr_Handler_Type;
   procedure Intr_Operation is
   begin
     --....
   end Intr_Operation;
begin
   loop
      The_Handler.Await;
      Intr_Operation;
   end loop;
end HW_Intr_Task;
```

```
protected type Intr_Handler_Type is;
   entry Await
private
   procedure Handle;
   pragma
     Attach_Handler(Handle,Ada.Interrupts.names.xxx);
   pragma Interrupt_Priority(32);
   Arrived:Boolean:=False;
end Intr_Handler_Type;
protected type body Intr_Handler_Type is
   entry Await when Arrived is
   begin
     Arrived:=False;
   end Await;
   procedure Handle is
   begin
     Arrived:=True;      --Reset HW Interrupt controller
  end Handle;
end Intr_Handler_Type;
```

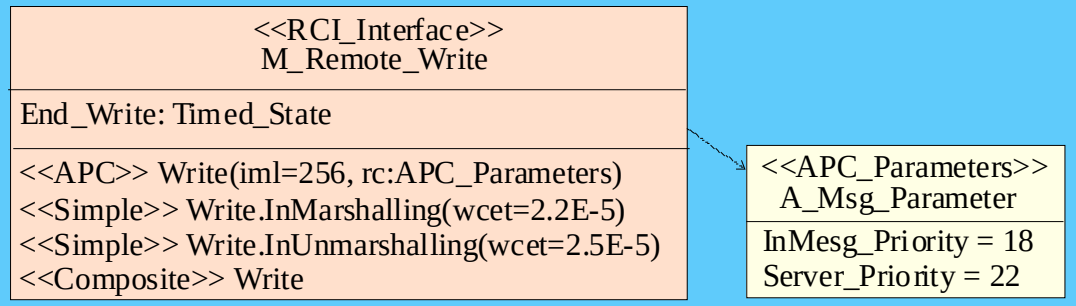
Ada constructs: hardware interrupt model



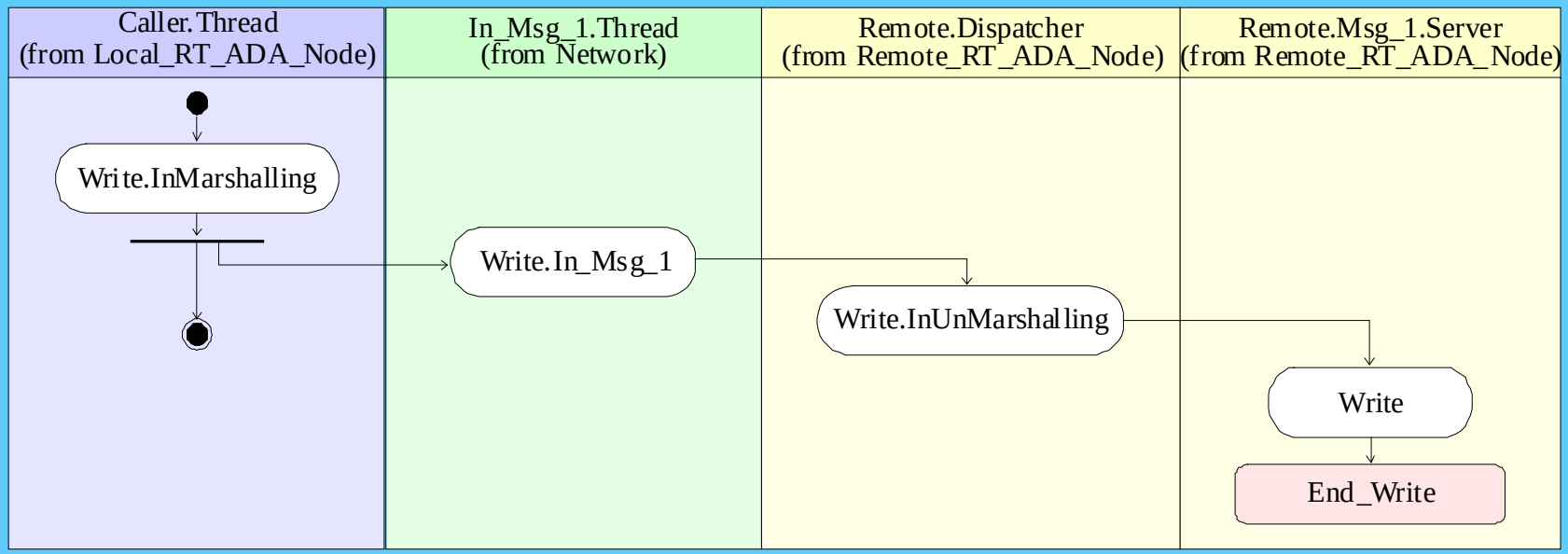
Ada constructs: an APC remote invocation

```
package Remote_Write is
   procedure Write(D: in Data_Type);
   pragma Remote_Call_Interface;
   pragma Asynchronous(Write);
end Remote_Write;
```

(a) Ada code of a remote call interface

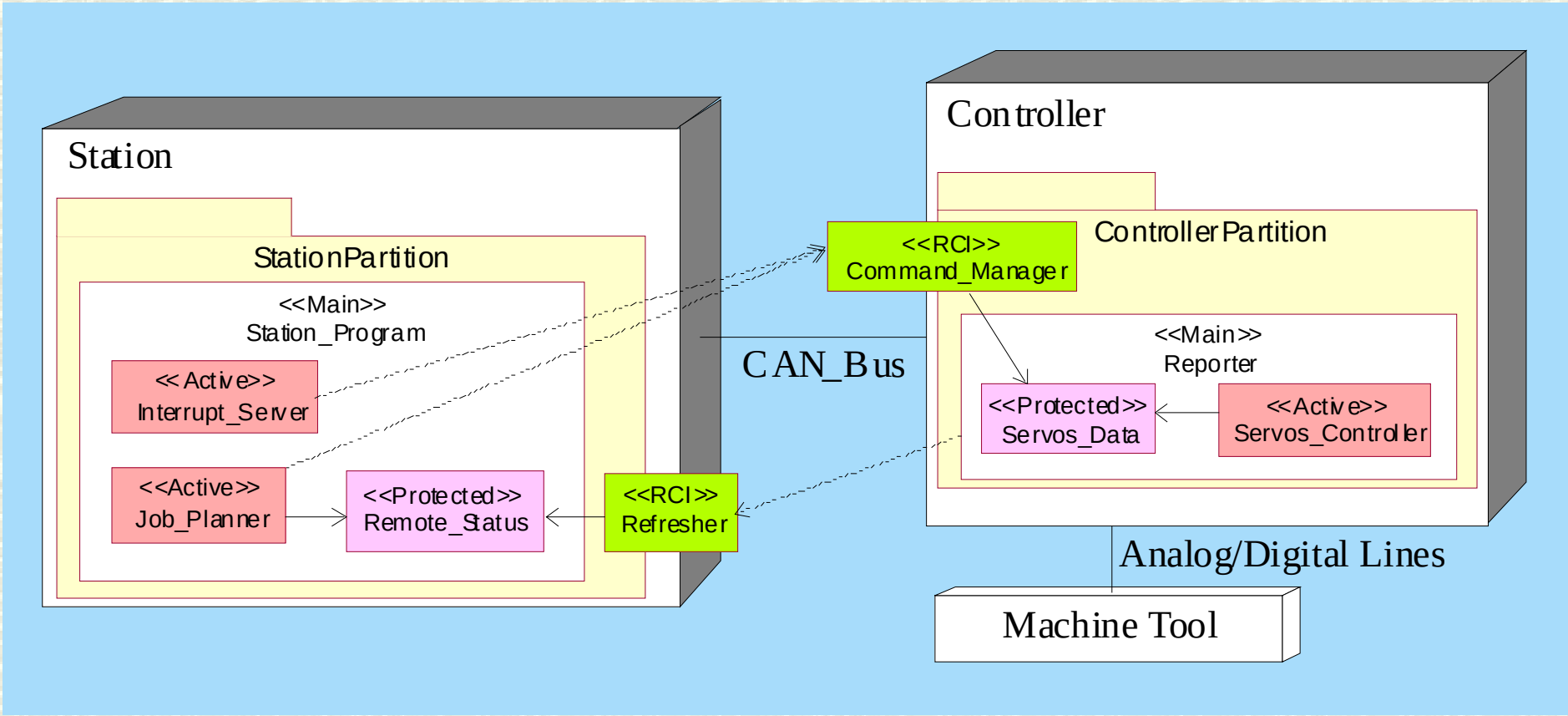


(b) MAST model of the remote call interface.

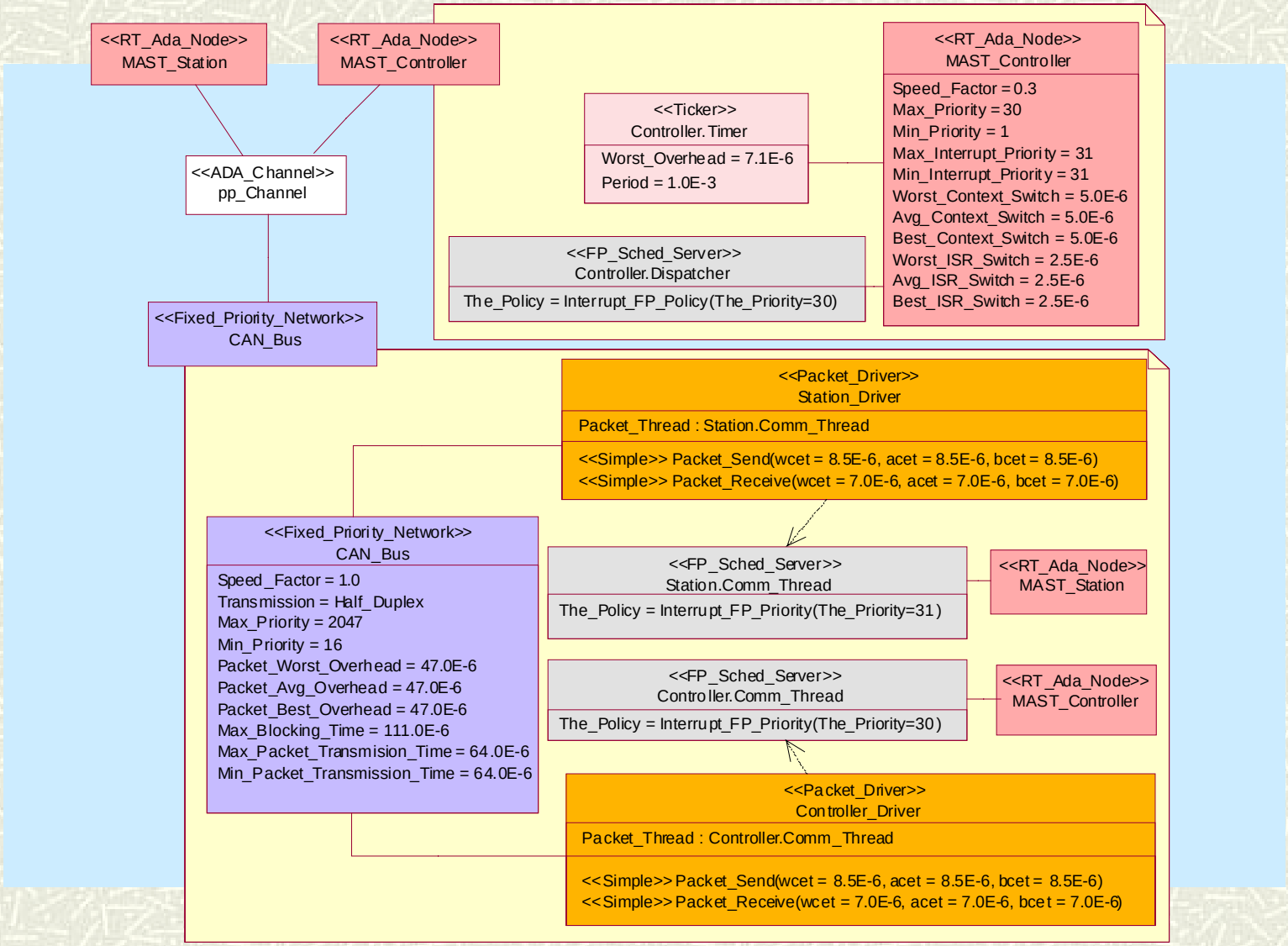


(c) Implicit activities diagram for APC Write.

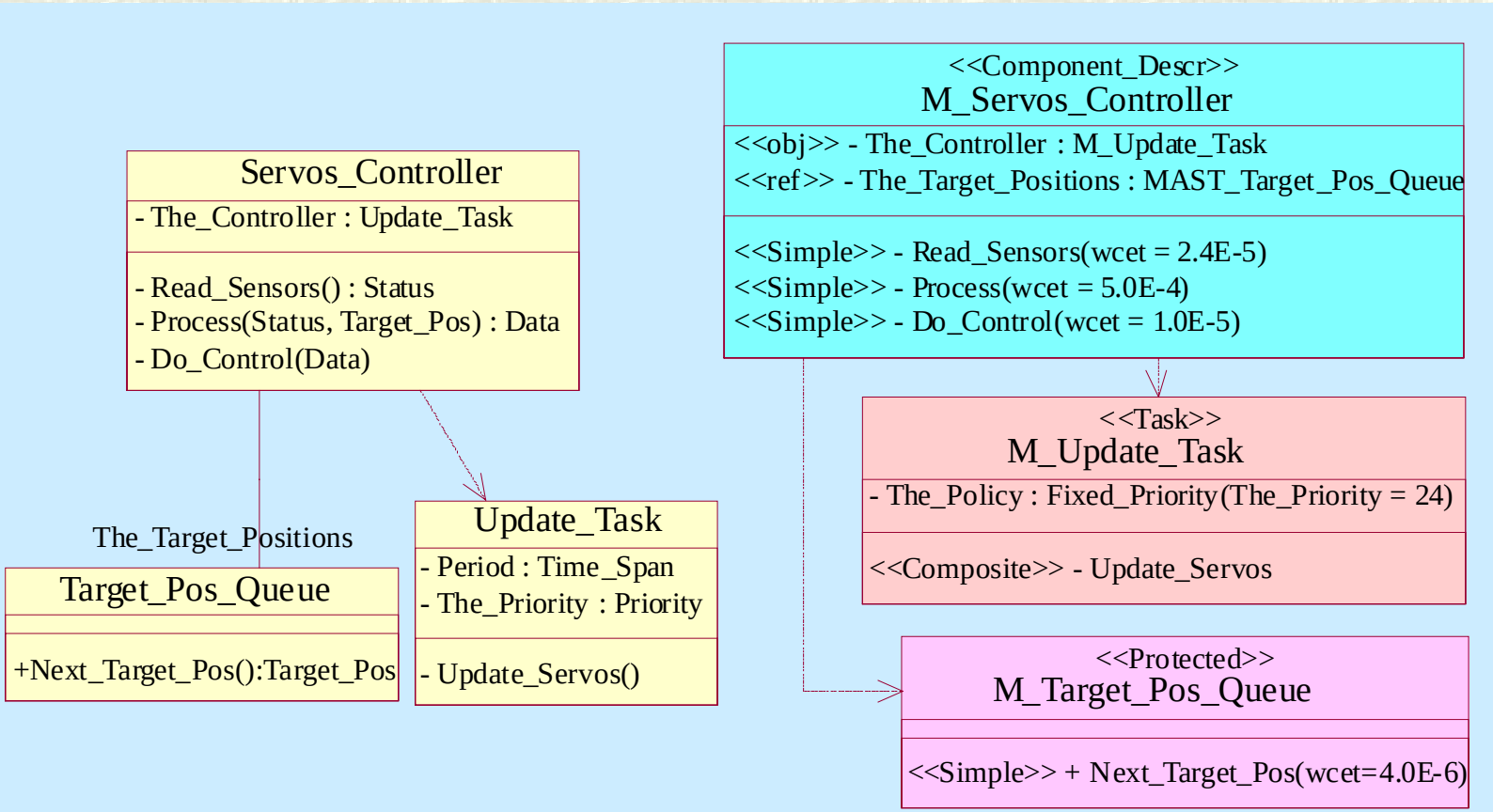
An Example: Teleoperated Machine Tool



Example: platform models



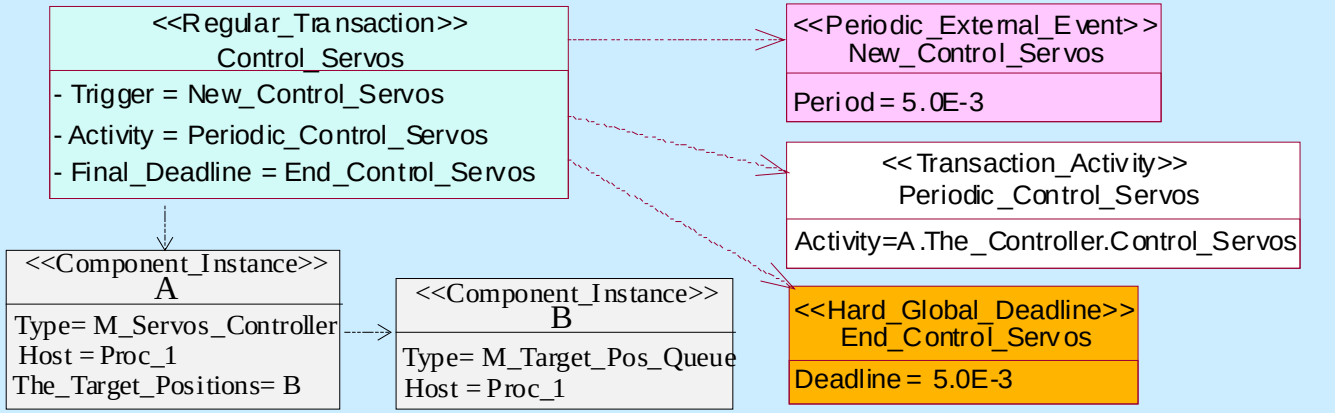
Example: logical components model



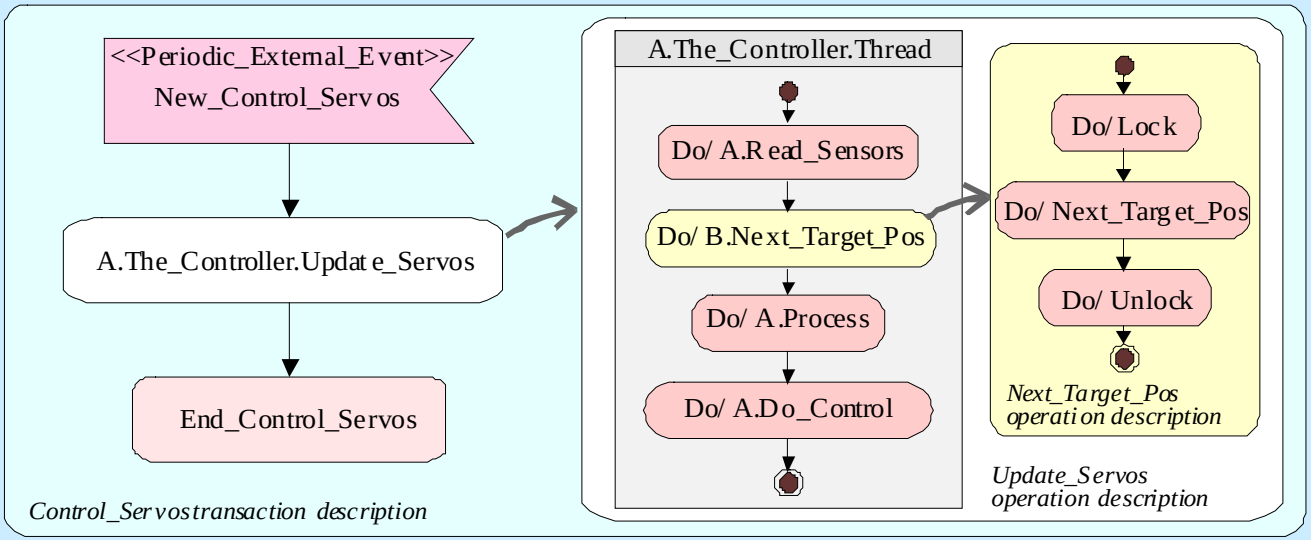
(a)Logical model of the control pattern.

(b)MAST real-time model of the control pattern.

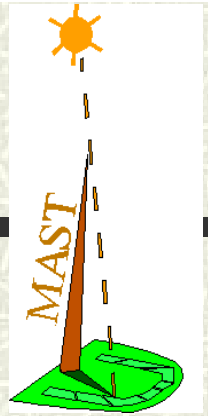
Example: model of a transaction



a) Declaration of the Control_Servos transaction.

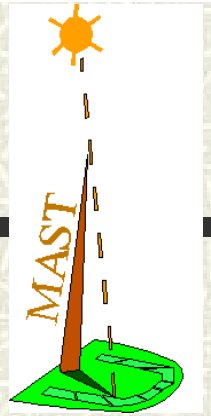


b) Description of the Control_Servos transaction and automatic recursive use of logical component models.



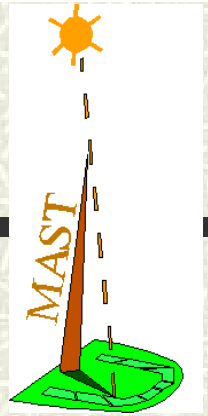
Example: schedulability analysis results

Transaction/Event	Slack	Worst response	Deadline
Control_Servos_Process	19.53%		
End_Control_Servos		3.833ms	5 ms
Report_Process	254.69%		
Display_Refreshed		34.156ms	100 ms
Drive_Job_Process	28.13%		
Command_Programmed		177.528ms	1000 ms
Do_Halt_Process	25.00%		
Halted		4.553ms	5 ms



Conclusions

- Advantages:
  - The methodology automates the application of well-known schedulability analysis techniques to distributed real-time systems written with Ada.
  - The methodology includes reusable real-time models of Ada components and patterns.
  - The designer is relieved of modeling de low-level artifacts introduced by Ada (context switches, background drivers, timers, remote invocations, etc.)
- Limitation:
  - Structures implemented with entries and guard conditions require a particular model for each usage pattern.



Complementary information about MAST

MAST is free code :

http://mast.unican.es