ETSIT UPM

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

AUTOMATIC GENERATION OF SCHEDULABILITY ANALYSIS - CONSISTENT CODE

STRAST

GROUP

dit UPM

EMILIO SALAZAR

esalazar@dit.upm.es

Table of contents

Motivation

- Code generation and schedulability analysis can be automatized.
- Avoid *blackboxed* tools and promote integration between tools
- Integrate code generation with schedulability analysis
- Create tools with MARTE's support
- Take advantage of Model Driven Architecture

Schedulability analysis

- All RTS must meet time constraints
- The profiles in OMG's MARTE standard address schedulability analysis in UML models
- Nowadays, there are few schedulability analysis tools that support MARTE's models directly and code generation
- Schedulability analysis tools are usually viewed as blackboxes which are developed isolated from the automatic code generators

Schedulability analysis (II)

- Writing time-deterministic code is very dependant of the programming language and the execution platform
- Automating the generation code will save time and reduce bugs and costs
- Modern programming languages include many non time-deterministic features (e.g. memory garbage collector, dynamic dispatching...)

Ada Ravenscar Profile

- ISO/IEC TR 24718:2005 is a subset of the Ada tasking features
  - » Addressed to Real Time and High Integrity systems
  - » Time-deterministic code
  - » Easier code certification

- Included in Ada 2005 as a *configuration pragma*

- Usual problems in concurrent programs:
  - » **Priority inversion**: high priority task blocked awaiting a resource used by a low priority task
  - » **Deadlock**: groups of tasks blocking each other
  - » **Livelock**: circular data dependencies between tasks
  - » **Missed deadlines**: The task fails to complete its work before its dead-line

Ada Ravenscar Profile (II)

- Tasks restrictions:
  - » Only static creation
  - » Fixed priority
  - » Interactions with others tasks only via protected objects
  - » Defined at library level
  - » No tasks hierarchies
  - » `select` and `abort` statements (rendezvous) are forbidden

- Protected objects restrictions:
  - » At most, one entry per protected object
  - » At most, one task queued at any time on that entry
  - » Simple entry barriers are forced

Ada Ravenscar Profile (III)

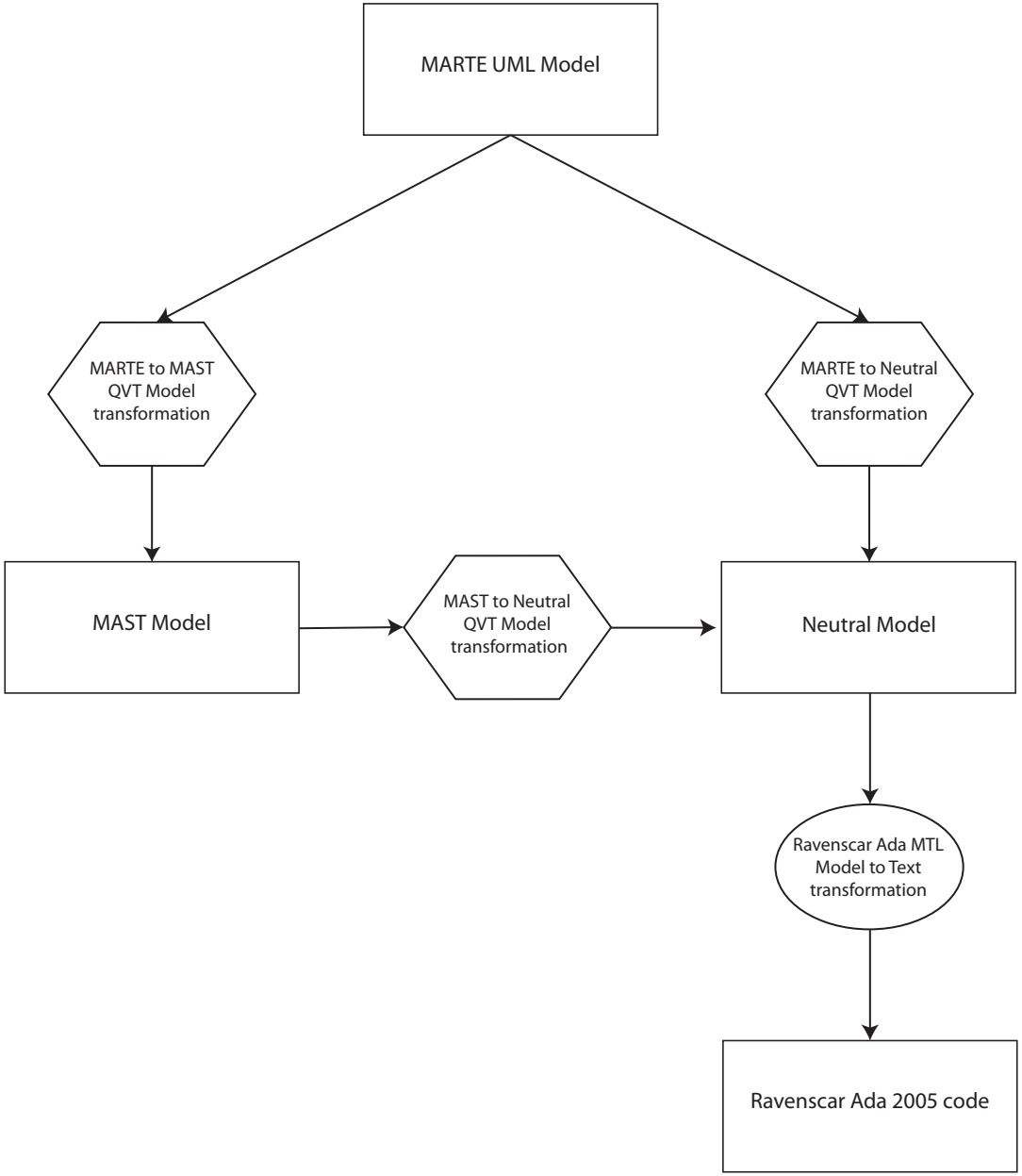
- Ceiling protocol and FIFO within priorities dispatching policy to assure the absence of deadlocks
- No dynamic memory from the standard storage pool
  - » Dynamic memory from user defined storage pools is allowed
- Only monotonic regular clock (`Ada.Real_Time`)
- Only absolute delays (`delay until`)

Ada code generator overview

- Advantages of the integration between analysis and generation tools:
  - » Avoids inconsistencies between model and code
  - » Code behaviors assumed by the schedulability analysis is known by the code generator and vice versa
  - » Platform specific code can be used by the code generator to assure consistency (e.g. WCET alarms)

- Code generation is split into two parts to achieve consistency:
  - » *Model to model* transformation: schedulability analysis model generation. Most of analysis decisions are taken here.
  - » *Model to text* transformation: Ada 2005 code generation

Ada code generator overview (II)



MARTE to Neutral transformation

- QVT Operational model to model transformation
  - » Input model: UML model annotated with MARTE
  - » Output model: *Plain* UML model

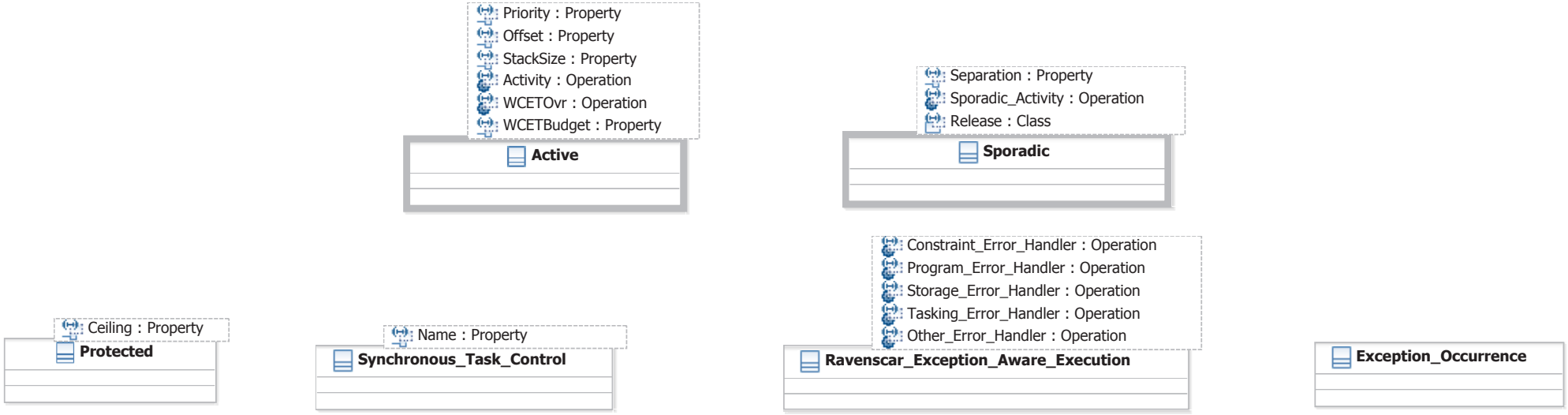
- MARTE input model must meet several constraints:
  - » (1 , ∞) <<saAnalisysContext>> packages
  - » (1 , ∞) <<GaResourcesPlatform>> classifiers
  - » (0 , ∞) <<SchedulableResource>> components or interfaces
  - » (0 , ∞) <<saSharedResource>> components or interfaces

- Extracts only code generation relevant information
  - » Period, jitter, priority, phase...
  - » Converts MARTE types to UML standard types
  - » Navigates among MARTE stereotypes

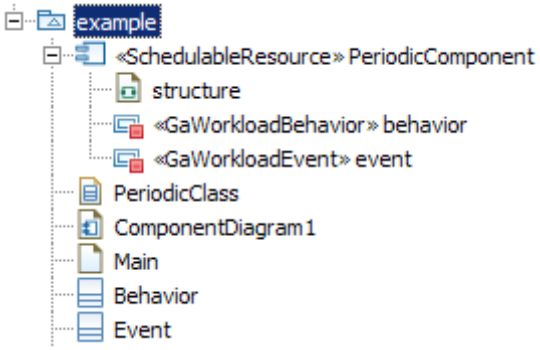
MARTE to Neutral transformation (II)

- Generates pattern based plain UML output model
  - » Only standard UML 2.1 types and data types
  - » Most common real-time patterns (periodic, sporadic, shared...)
  - » Generic UML classes with direct translation to Ada generic packages

- Avoids stereotypes and non-standard UML types because they are not fully supported in MTL

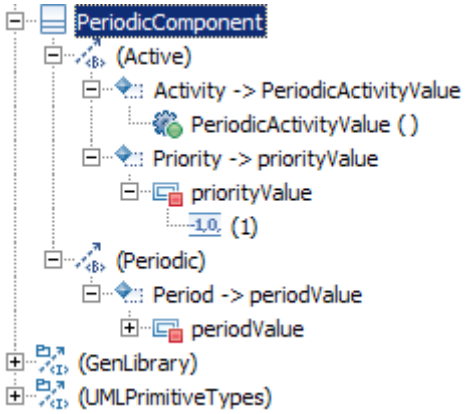


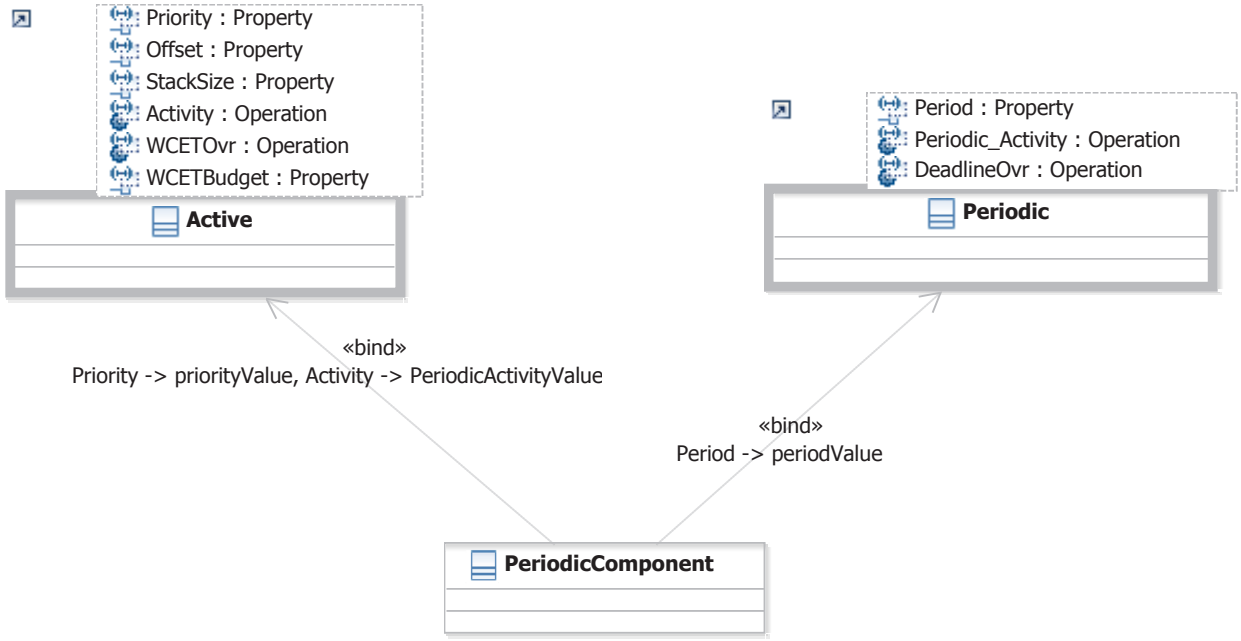
MARTE to Neutral transformation (III)



UML Classifier	MARTE Stereotype	Neutral Model
Component	SchedulableResource	**Active**
Component	saSharedResource	**Protected**
Interface	SchedulableResource	**TaskInterface**
Interface	saSharedResource	**ProtectedInterface**

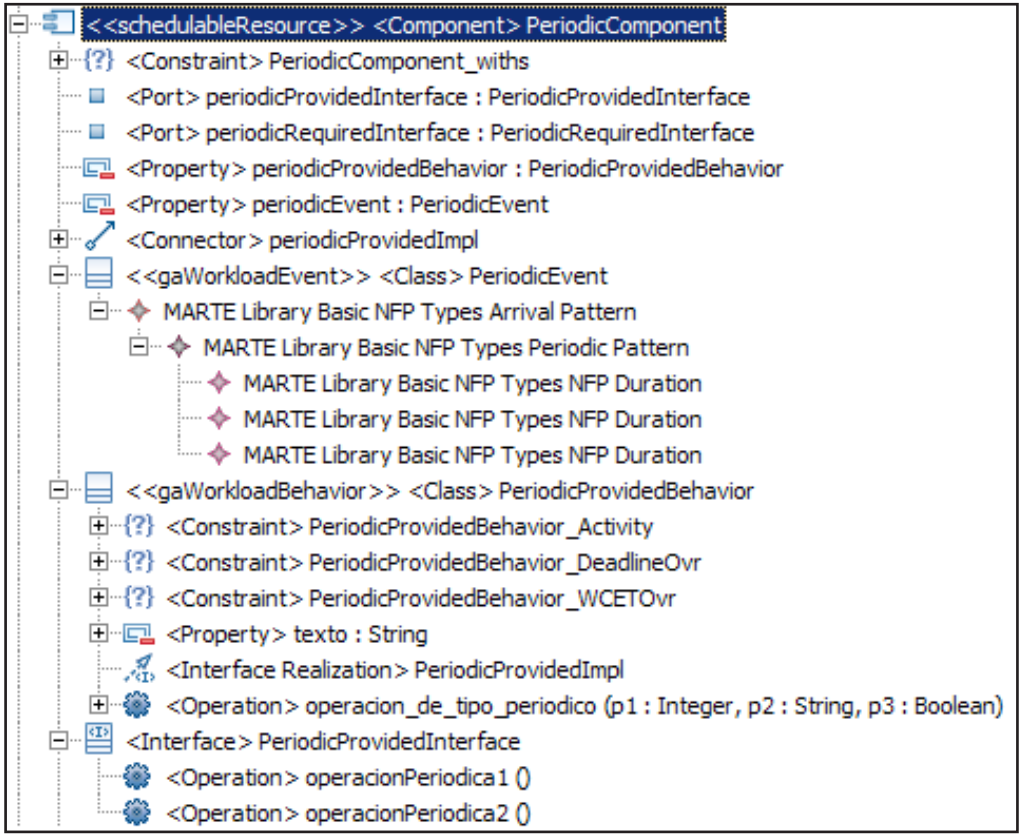
QVT transformation: MARTE to Neutral



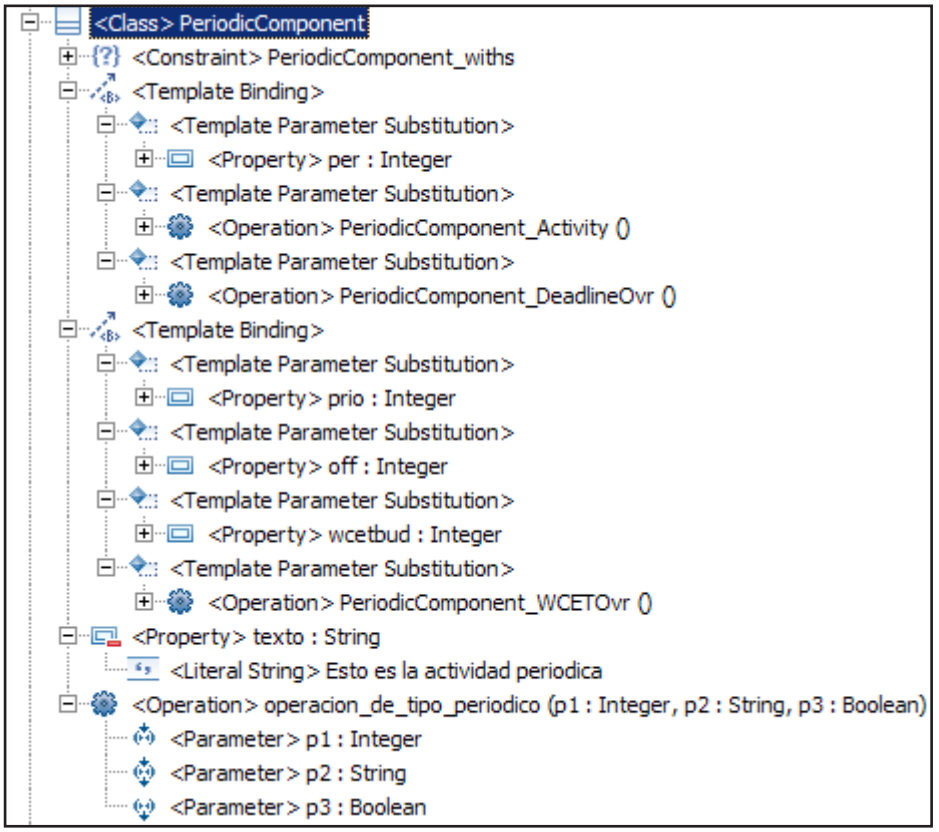


MARTE to Neutral transformation (IV)

MARTE Model



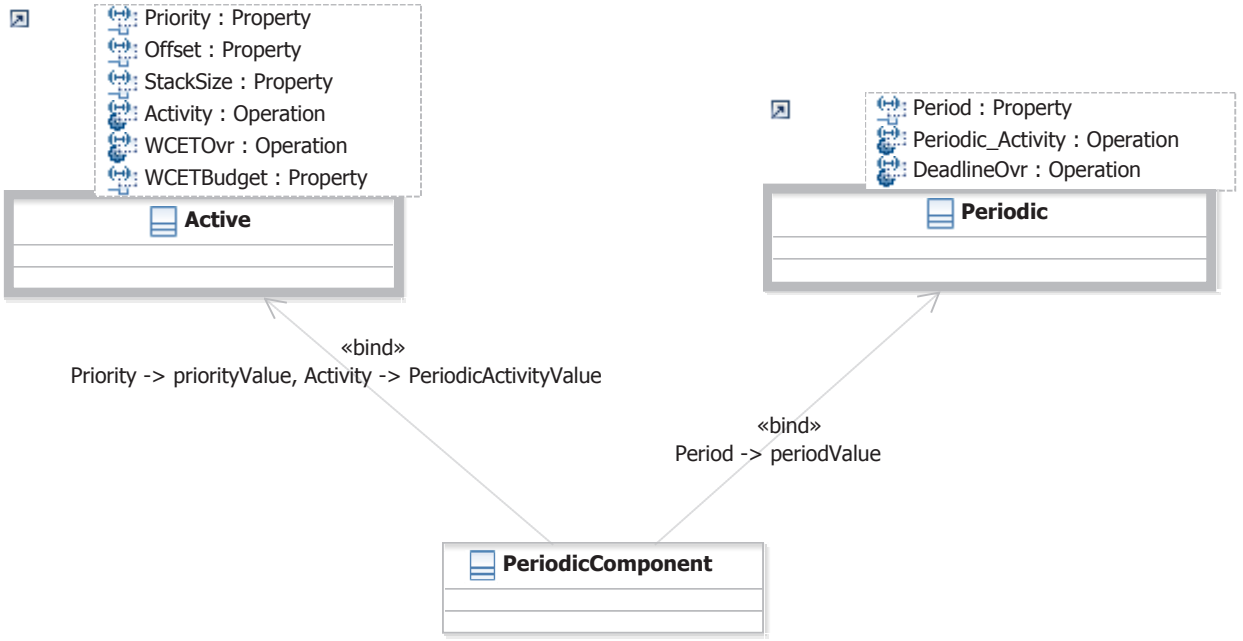
Neutral Model



Neutral to Ada generation

- MOF2Text Language (MTL) transformation
  - » Input: UML model
  - » Output: Ravenscar Ada 2005 code
- Code generation is based on a library of Ada generic packages
- Each UML class in the library has one Ada generic package associated
- The MTL generator must *only* instantiate the correct Ada package with the given parameters

Neutral to Ada generation (II)



MTL generation: Neutral to Ada

```
package PeriodicComponent_package is
      new wcet_periodic_tasks (
            Priority => 1,
            Period => 1000,
            Offset => 500,
            Periodic_Activity => periodiccomponent_activity,
            Deadline_Ovr_Handler => periodicComponent_DeadlineOvr.Hndl'Access,
            WCET_Budget => 10,
            WCET_Ovr_Handler => PeriodicComponent_WCETOvr.Hndl'Access);
```

Note: Red text is data extracted from the model and fitted by MTL generator

MARTE's issues for generating code

- UML and MARTE are huge, there are many ways to say the same thing
  - » A subset with only one way to represent the same thing:
    - Simpler automatic code generation
    - Simpler traceability (in both ways) management

- UML (and MARTE) are very targeted to the object-oriented paradigm
  - » Complicated to make UML or MARTE designs without OO elements
  - » Non-OO code is usually easier to analyze

- MARTE defines its own types and data types, even those that are already defined in UML (i.e. integer, string, boolean)

MARTE's issues for generating code (II)

- Support for exceptions handling
- Support for handling deadline and WCET overruns

Conclusions

- Schedulability analysis and code generation integration makes consistency between models and code easier

- Schedulability analysis and code generation automation
  - » Reduces RTS costs
  - » Increases RTS reliability
  - » Reduces RTS development time

- MARTE's support for code generation could be improved

- Ada Ravenscar subset allows to assume code behavior, which make easier the code generation

ETSIT UPM

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

Thank you for your attention!

STRAST

GROUP

dit

UPM