A Model-Based Framework for Developing Real-Time Safety Ada Systems

Emilio Salazar
Alejandro Alonso,
Miguel A. de Miguel
Juan A. de la Puente

dit
UPM
STRAST

Universidad Politécnica de Madrid (UPM)

Overview

- Model-based development for embedded real-time systems
  - analysable temporal behaviour
- Use standards
  - UML2 MARTE for higher-level models
  - Ada & Ravenscar profile for code
  - QVT, MTL for transformations
- Provide automatic code generation
- Not limited to a specific toolset
  - can be used with RSA, Eclipse, Papyrus, ...

UML MARTE

- UML2 profile for modelling properties of real-time embedded systems
- Large and complex
  - foundations
    - basic concepts for modelling RTES
  - specializations
    - analysis
    - design

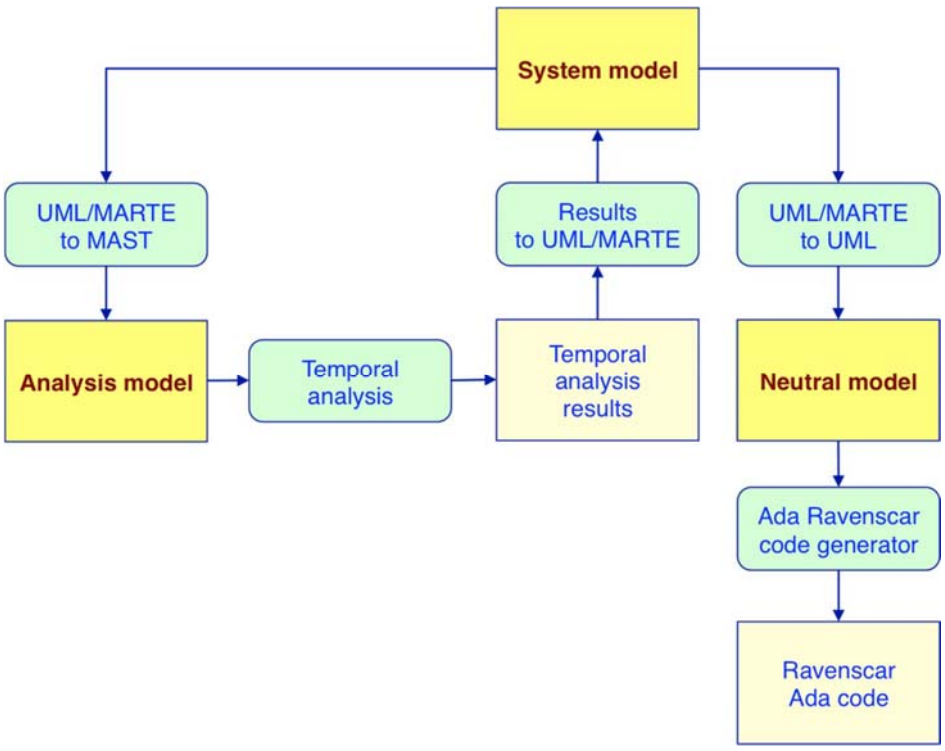
Modelling elements for RTES

- In order to support an analysable temporal behaviour models must be restricted
  - Input events
    - periodic, sporadic
  - Actions
    - response to events
  - Precedence constraints
    - define end-to-end flows with deadlines
  - Resources
    - active resources
      - ✓e.g. processors, communication links
    - passive resources
      - ✓e.g. shared data objects

MARTE profiles

- GQAM (Generic Quantitative Analysis Modelling)
  - common modelling abstractions
    - GaWorkloadEvent
    - GaScenario
    - GaStep
    - GaResourcesPlatform
- SAM (Scheduling Analysis Modelling)
  - refined abstractions for temporal analysis
    - SaExecHost
    - SaCommHost
    - SharedResource
    - Scheduler
    - SecondaryScheduler

Model-Based Framework



System model

- Global model of system using UML MARTE
  - starts as platform-independent model (PIM)
    - only behaviour, no resources
    - end-to-end timing requirements
  - evolves to platform-specific model (PSM)
    - resources added
    - WCET estimates, later replaced with measured values
- Limited set of class archetypes
  - periodic
  - sporadic
  - protected
  - passive

MARTE profiles for class stereotypes

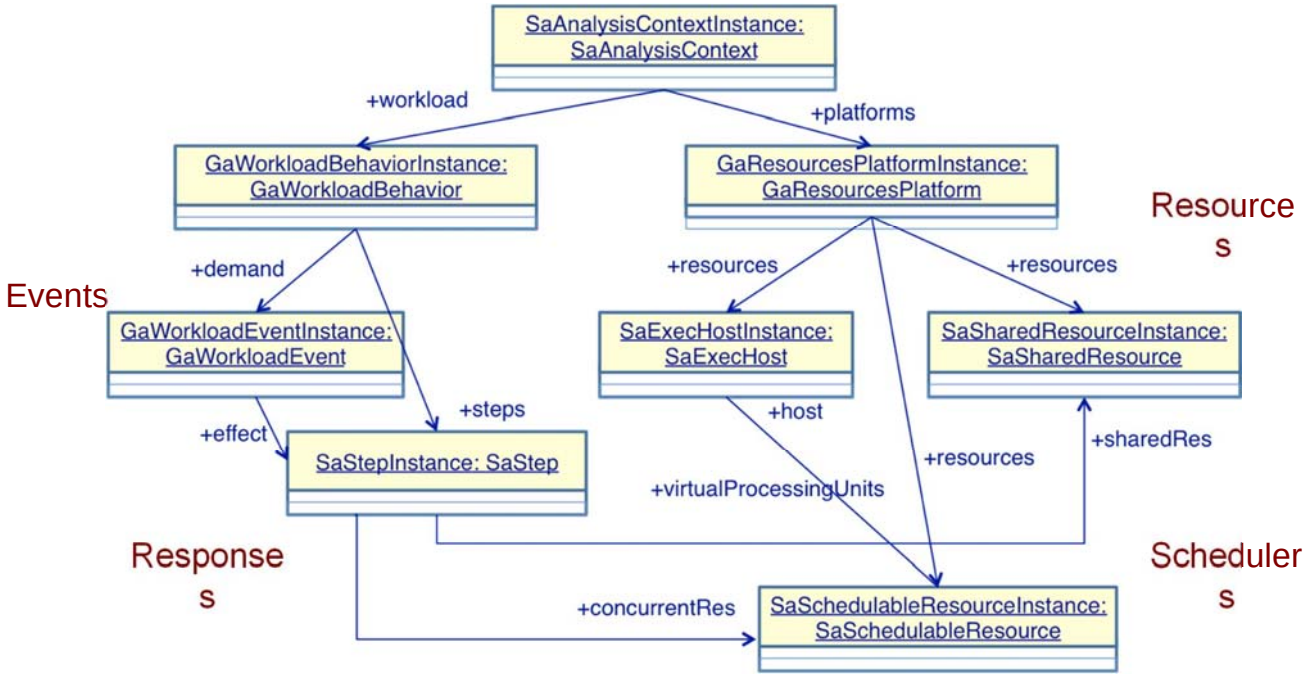
- Periodic and sporadic classes
  - GQAM::GaWorkloadEvent
    - arrival pattern, deadline
  - GRM::Schedulable- Resource
    - provides scheduling details
      - ✓FPPS assumed by default
- Protected classes
  - GRM::MutualExclusionResource
    - objects with mutually exclusive access
      - ✓ICPP assumed by default
- Passive classes
  - no MARTE stereotypes

Analysis model

- Provides temporal analysis results
  - worst-case response times & other data
- Based on MAST toolset
  - open-source, widely available
  - sophisticated analysis techniques
- Automatically generated from system model
- Feedback to system model
  - transformation tools

Analysis scenarios

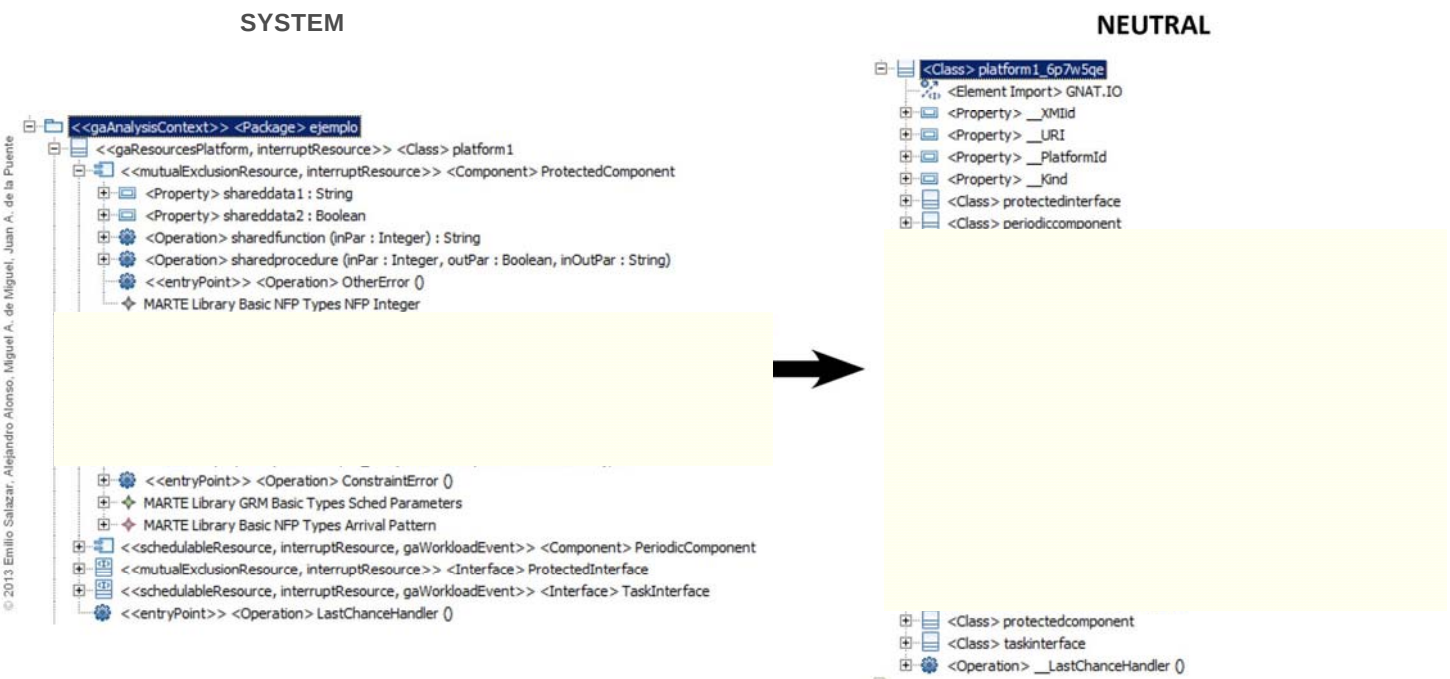
- Defined with MARTE SAM::SaAnalysisContext
- Translated to MAST language



Neutral model

- Aimed at code generation
- Plain UML, no MARTE annotations
- Generated from system model
- A few basic concepts
  - simplicity: include only what is needed
  - independence from programming language
  - traceability from system model to generated code
- Common real-time patterns
  - based on system model stereotypes
- Additional annotations
  - WCET values, handler definitions

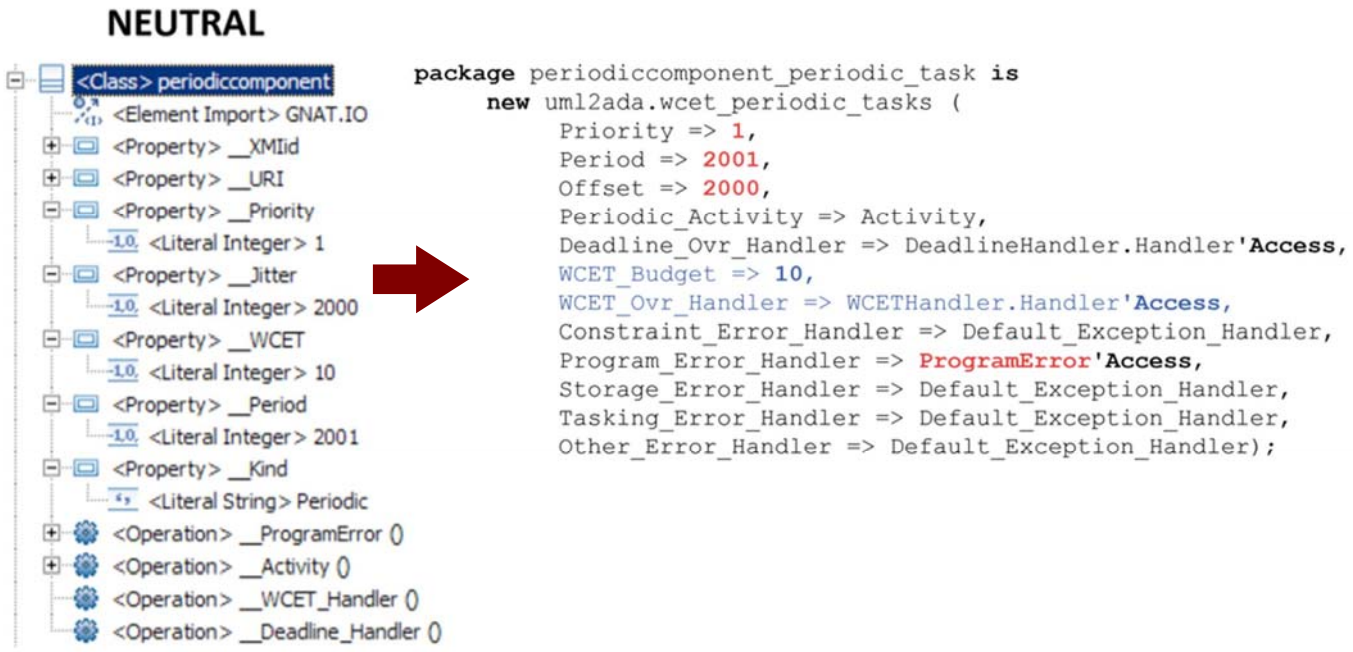
Example



Code generation

- Code generator built with standard tools
  - QVT, MTL
- Generates Ada / Ravenscar code
  - Real-Time Java generator also available
- Code generation based on tasking patterns
  - periodic, sporadic, protected components
- Functional code from other tools can be easily integrated

Example



The UPMSat2 ADCS subsystem

- Micro-satellite mission developed at UPM
- Attitude control software
  - measure attitude angles with magnetometers
  - other sensors also available (solar cells)
  - determine correcting torque direction and magnitude
  - set voltage applied to magnetorquers

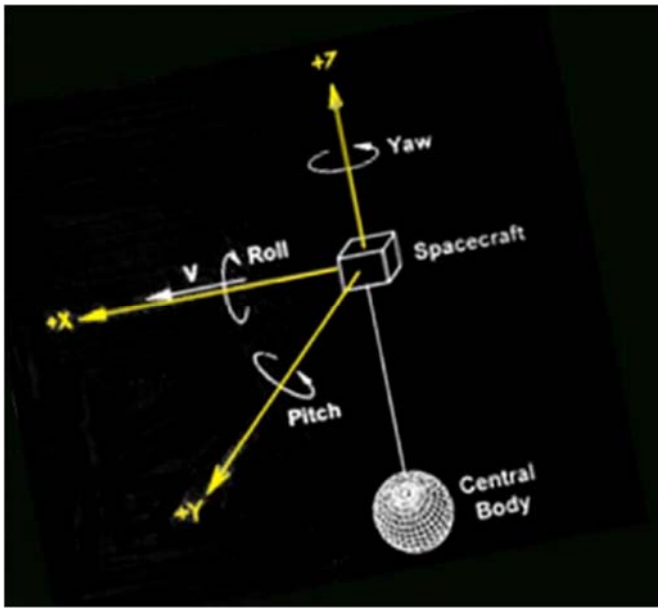
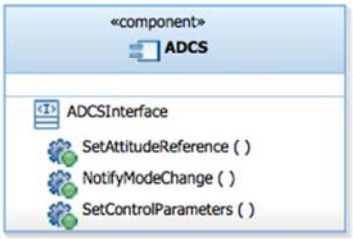
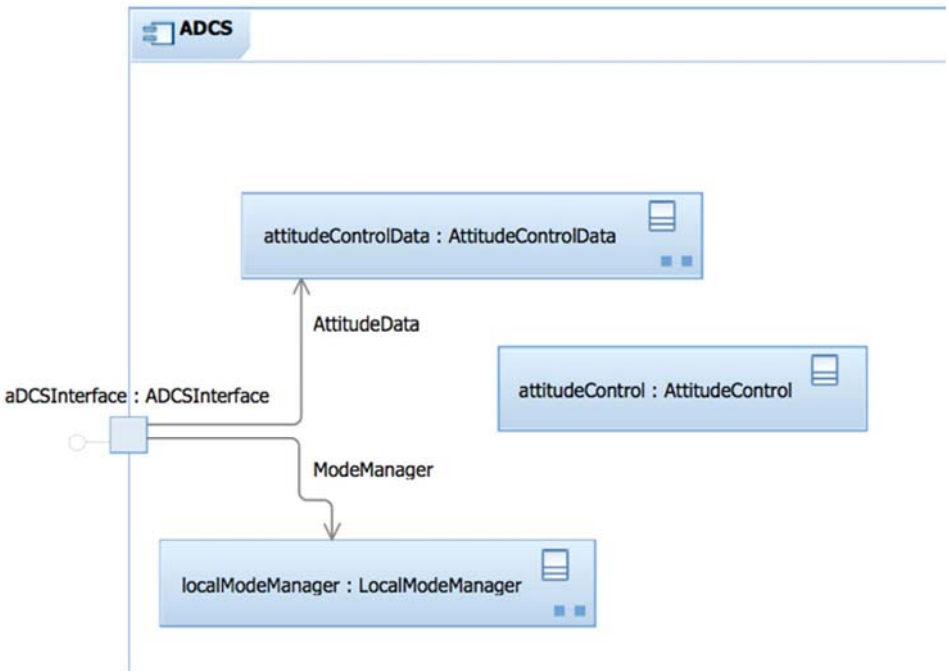


Figure by Assal Farrahi

ADCS component





Code generation for components

- Interface implemented as Ada package
  - operation specifications
  - body redirects to internal classes
- Internal classes implemented as child packages
  - using real-time archetypes

Example

```
with ADCS.BasicTypes; use ADCS.BasicTypes;
package ADCS.Interfaces is
   procedure NotifyModeChange     (mode : in Mode_Type);
   procedure SetAttitudeReference (ref  : in Reference_Type);
   procedure SetControlParameters (conf : in Configuration_Type);
end ADCS.Interfaces;
```

```
with  ADCS.LocalModeManager;
with  ADCS.AttitudeControlData;
package body ADCS.Interfaces is

   procedure NotifyModeChange (mode : in Mode_Type) is
      begin    ADCS.LocalModeManager.LocalModeManager.SetMode  (mode);
   end  NotifyModeChange;

   procedure SetAttitudeReference (ref : in Reference_Type) is
   begin
      ADCS.AttitudeControlData.AttitudeControl.SetActitudeReference(ref);
   end  SetAttitudeReference;

   procedure SetControlParameters (conf : in Configuration_Type) is
   begin
     ADCS.AttitudeControlData.AttitudeControl.SetControlParameters(conf);
   end  SetControlParameters;
end  ADCS.Interfaces;
```

Internal classes

Code generation for classes

- Classes implemented as Ada packages
  - Ravenscar-based code archetypes used
  - defined as generic Ada packages
    - originally derived from HRT-HOOD
    - later refined in UML terms

Example

```
with Ada.Real_Time.Timing_Events; use Ada.Real_Time.Timing_Events;
with GNAT.IO;                     use GNAT.IO;
with Ada.Exceptions;              use Ada.Exceptions;
with UML2Ada.Exceptions;          use UML2Ada.Exceptions;
with UML2Ada.Periodic_Task;

package ADCS.AttitudeControl is
   Priority : constant := 8;
   Period   : constant := 9854;       -- milliseconds
   Offset   : constant := 234000000;  -- milliseconds

private
   procedure Control;
   procedure Start_Control;

   protected DeadlineHandler is
       procedure Deadline_Error_Handler (Event : in out Timing_Event);
   end DeadlineHandler;

   procedure Constraint_Error_Handler (e : in Exception_Occurrence);

   package ADCS.AttitudeControl_Thread is new
     UML2Ada.Periodic_Task (
       Priority, Period, Offset,
       Activity => Control,
       Initialization            => Start_Control,
       Deadline_Ovr_Handler      => Deadline_Error_Handler'Access,
       Constraint_Error_Handler  => Constraint_Error_Handler'Access
        Program_Error_Handler    => Default_Exception_Handler,
       Storage_Error_Handler     => Default_Exception_Handler,
       Tasking_Error_Handler     => Default_Exception_Handler,
       Other_Error_Handler       => Default_Exception_Handler);

end ADCS.AttitudeControl;
```

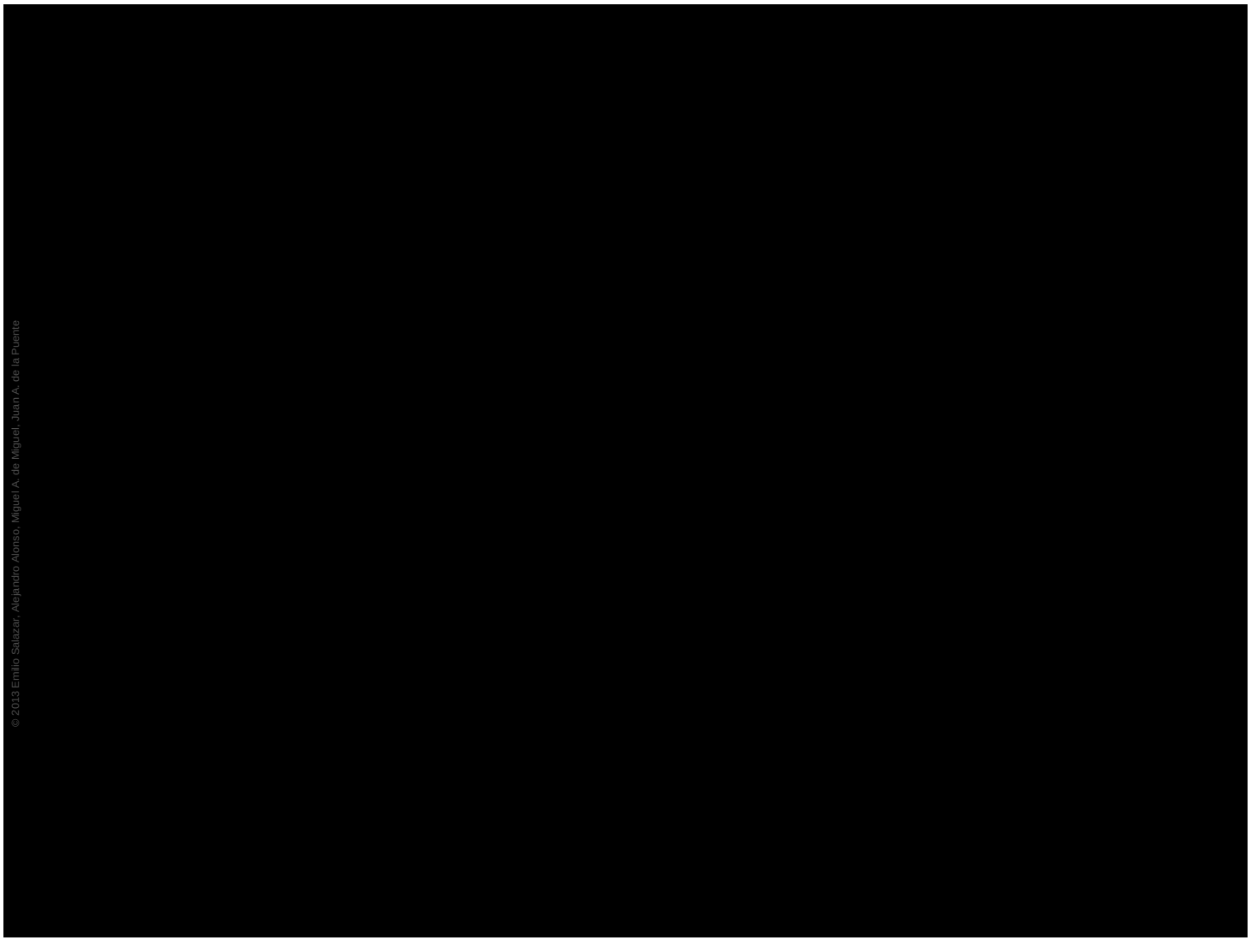

Conclusions

- Framework aligned with international standards
- Tested on Rational Software Architect
  - migration to Eclipse planned
- Transformation tools available at
  - http://www.dit.upm.es/str/projects/ERMA/

Current and future work

- Complete work on timing analysis round-trip
- Complete integration of functional code from Simulink® and other tools

- Extend to partitioned systems
  MultiPARTES, FP7 ISP 287702



© 2013 Emilio Salazar, Alejandro Alonso, Miguel A. de Miguel, Juan A. de la Puente