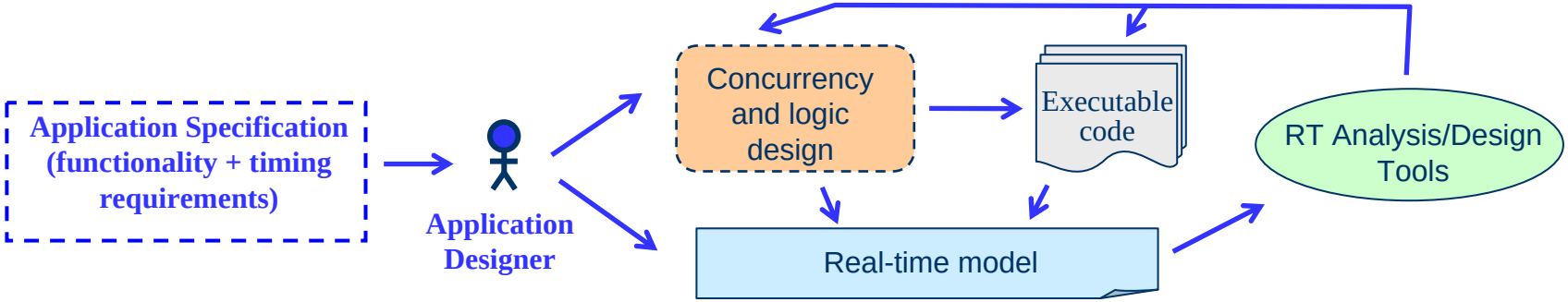
Modular real-time models for complex systems

Patricia López Martínez and José M. Drake

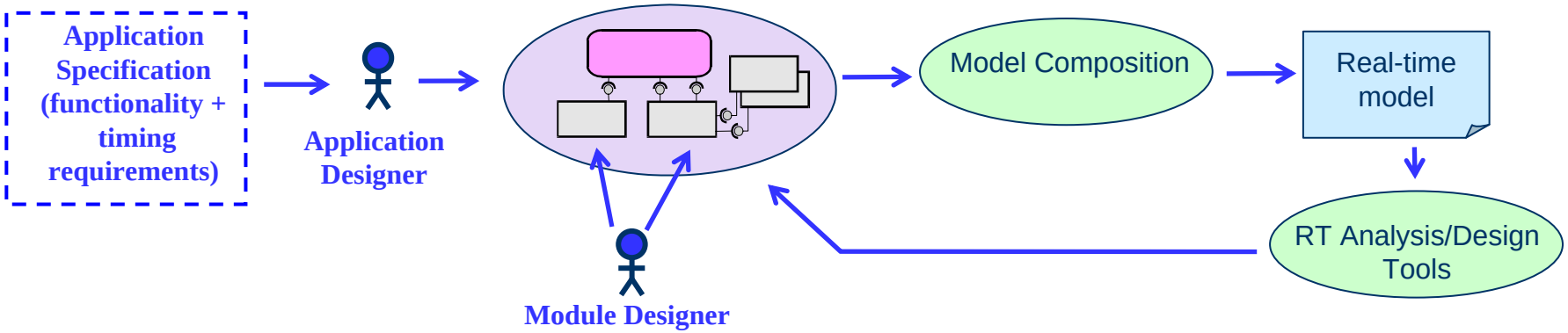
Grupo de Computadores y Tiempo Real
Universidad de Cantabria, Spain

Classical real-time design vs new design paradigms

- Classical real-time design:
  - The designer knows all the details about the platform and the applications
  - This is the scenario considered in many modelling and analysis environments (MAST, MARTE, …)



- New paradigms of real-time design for complex systems:
  - Component-based, legacy code, heterogeneous systems, etc…
  - The designers do not know the internal details of all the elements that form the system



Modular and composable real-time models

- Our approach for modelling complex systems consists in:
  - Associating to each reusable software/hardware module a real-time model containing all the information about its temporal behaviour required to evaluate the timing behaviour of any application in which the module may be used
  - Generating the real-time model of a system as a composition of the models of all the elements that form it

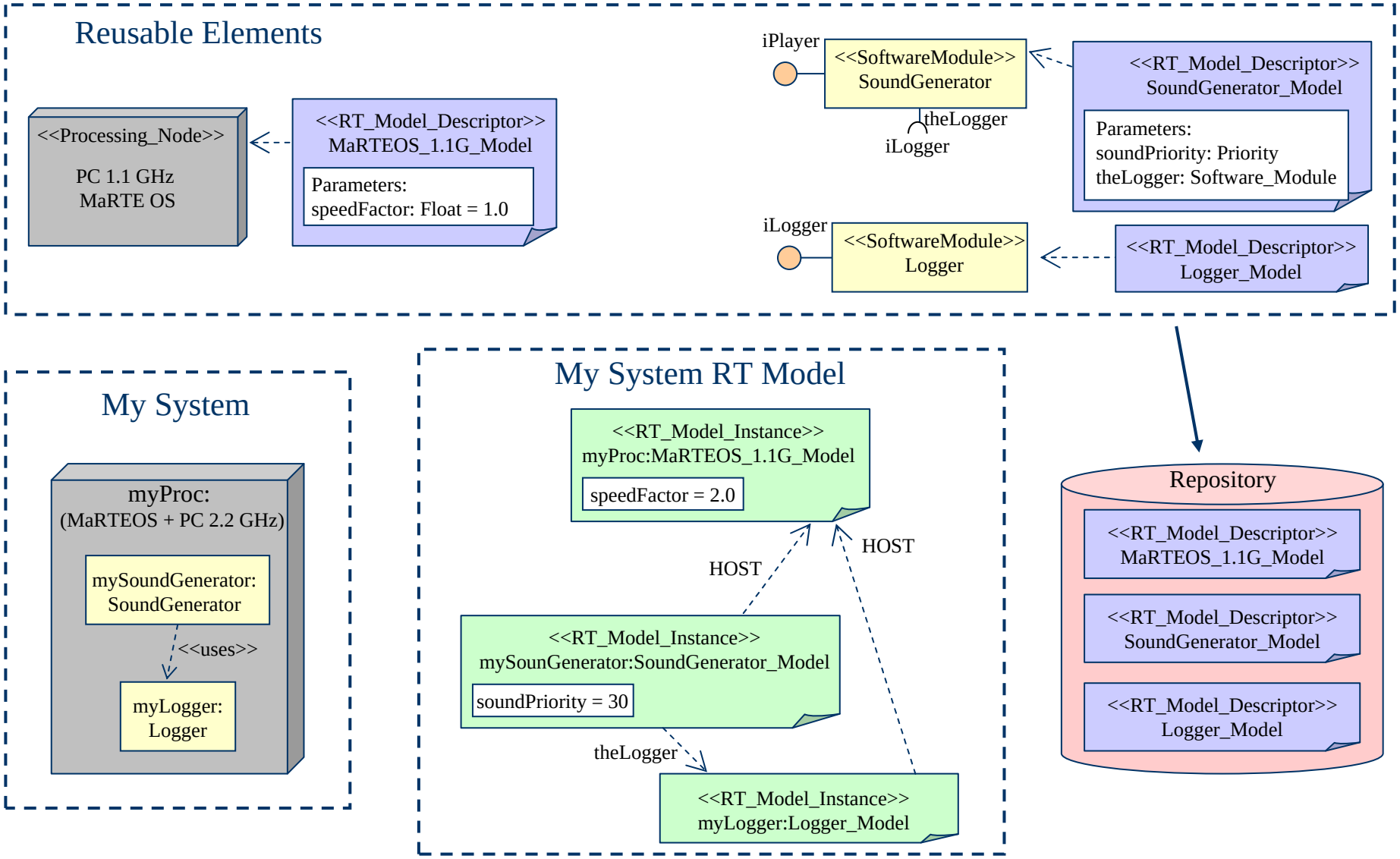
- For the success of the composition process, the real-time model of a reusable software/hardware module must be:
  - Complete: Information required to generate the real-time model of any application in which the module can be used
  - Reusable: Independent of the application in which it is used
  - Composable: With the models of other modules that interact with it

- The temporal behaviour of a software module depends on:
  - The characteristics of the execution platform
  - The behaviour of other software modules
  - Availability of the platform resources: Workload

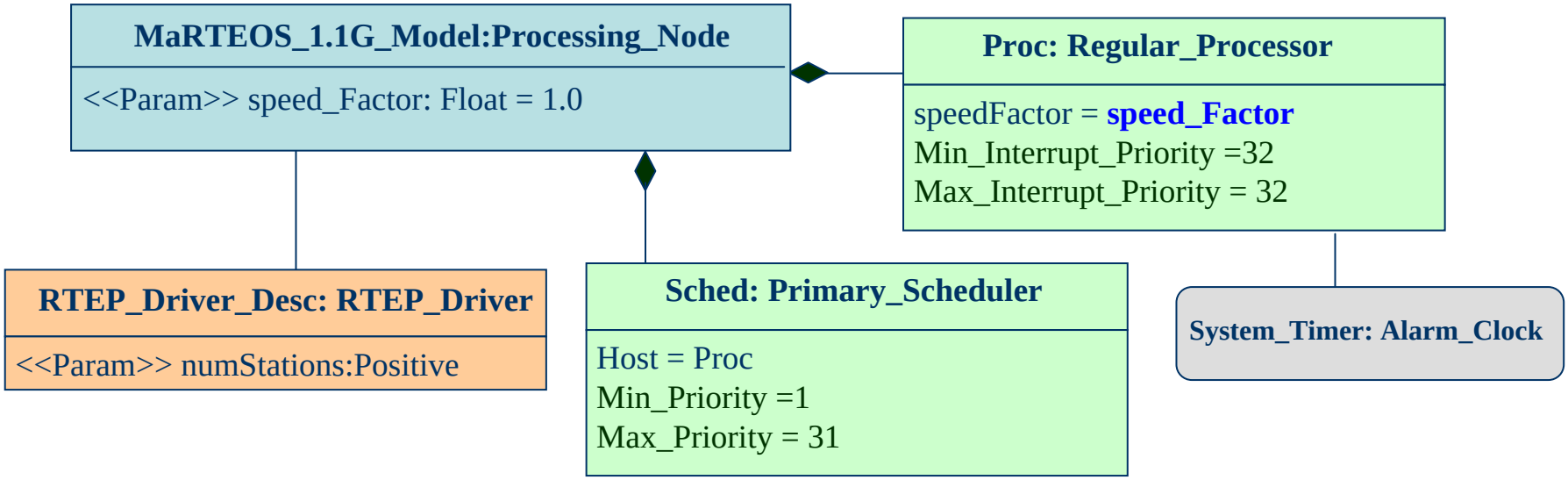
Mod-MAST: Modular MAST

- Mod-MAST is an extension of MAST that provides modularity and composability:
  - It uses the set of modelling primitives defined in MAST but they can be parameterized
  - It defines container modelling elements to formulate the real-time model of reusable:
    - Platform elements: Processing Node, Communication Network and Communication Service.
    - Application elements: Software Module.
  - It is based on the concept of Model Descriptor vs Model Instance
    - Model Descriptor:
      - Parameterized template that describes the temporal behaviour of a module independently of the application
      - Instantiation dependent characteristics formulated as parameters or references to other models
    - Model Instance:
      - Complete analyzable model of an instance of a module in a concrete real-time situation
      - Obtained by assigning concrete values to the parameters and the unresolved references of the corresponding descriptor

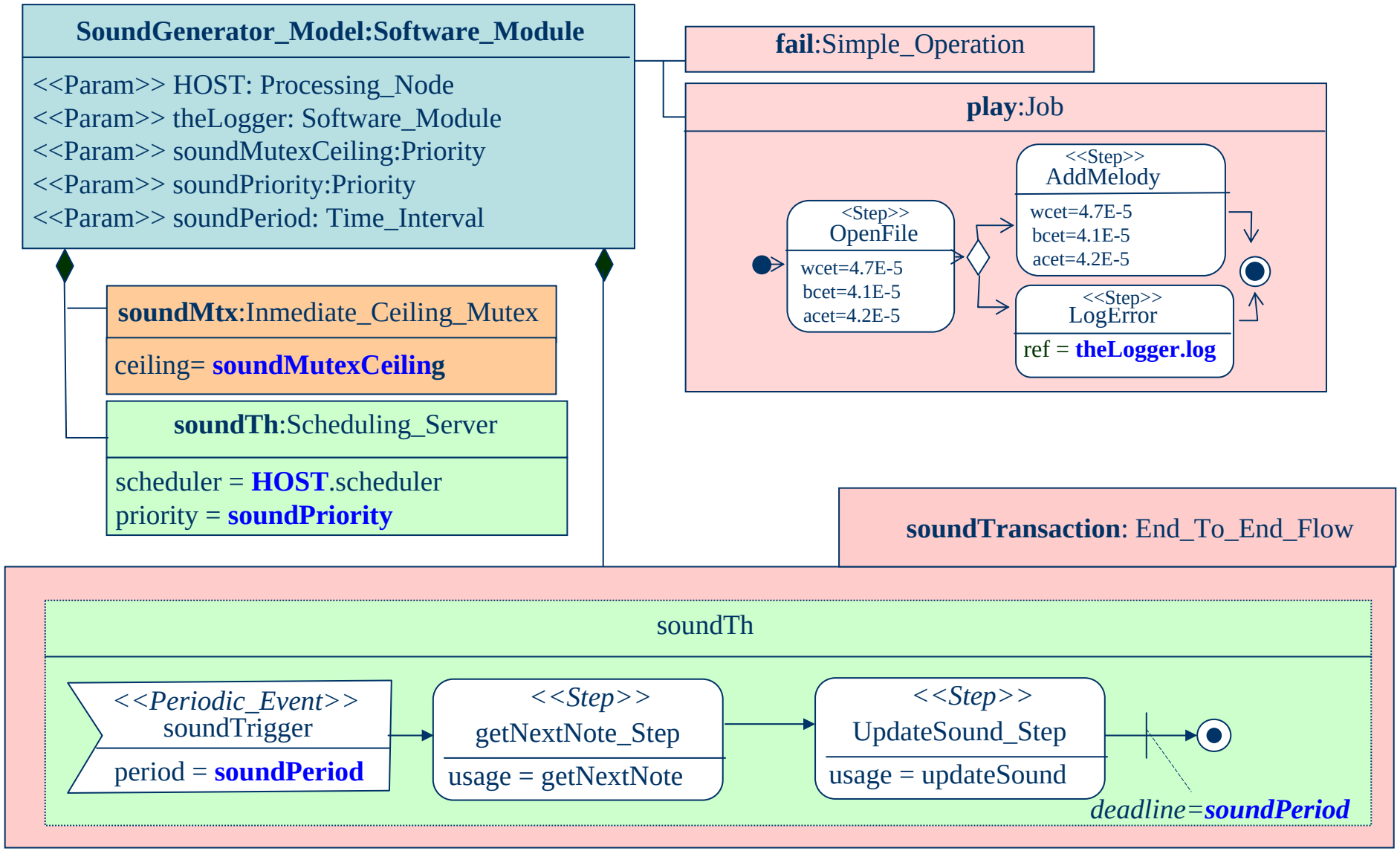
Descriptor vs Instance



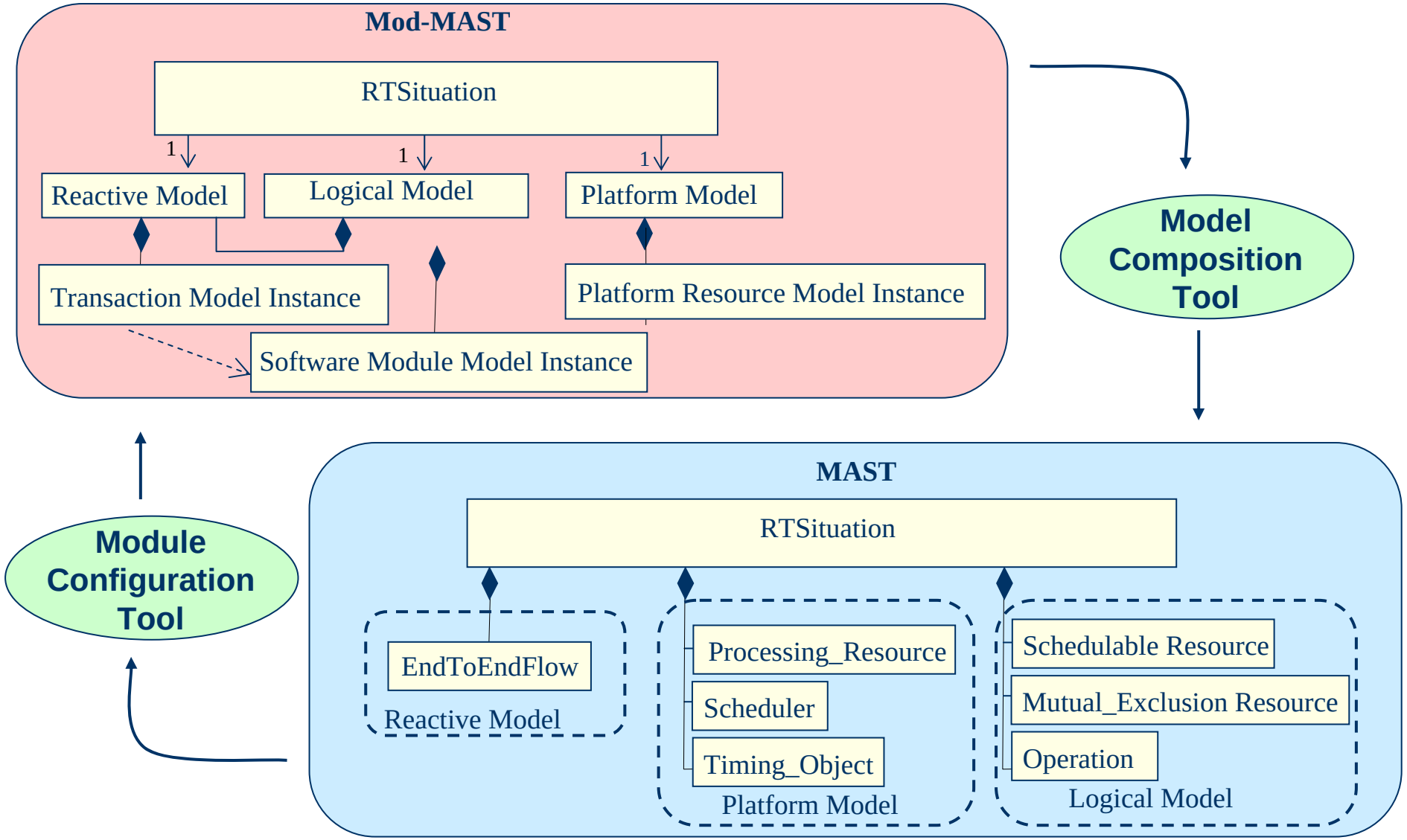
Example of Processing_Node descriptor



Example of Software_Module descriptor

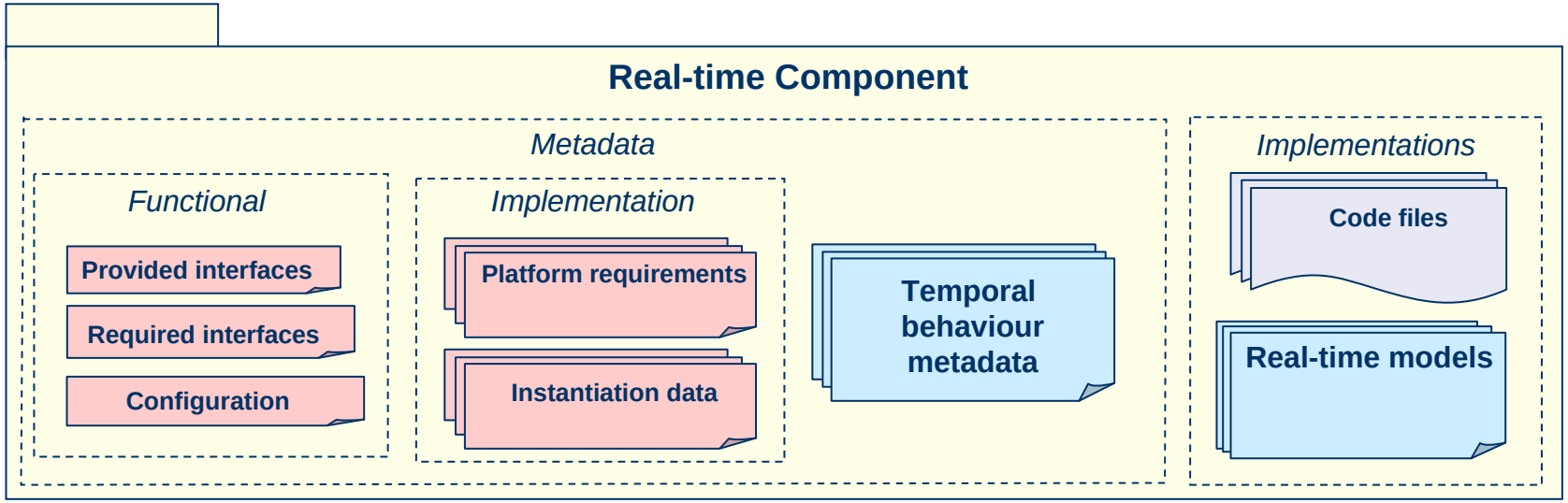


From Mod-MAST to MAST



CBSE-MAST: MAST for component-based systems

- CBSE-MAST: Adaptation of Mod-MAST to component-based systems
- New container elements:
  - Software Component: Provided and required ports
  - Software Connector
- The real-time model descriptor as part of the information provided by the component
  - Opacity: The real-time model is managed and configured through metadata provided with the component package => RT-D&C



Open lines for future work

- Opaque management of the reusable models of the components
  - Standard formulation for the external view of the real-time model (as IDL for the functional view).
  - The external view includes the information required to:
    - Evaluate if two components are composable from the rt point of view
    - Adapt the temporal behaviour of a component to a concrete application
    - Define the workload that a component can generate in an application

- Real-time models formulated independently of the analysis tools, using MARTE, but:
  - MARTE is defined at “instance” level => Parameterization is needed for reusability and composability
    - ¿MARTE variables?
  - MARTE, specially the SAM chapter, is oriented to formulate the temporal behaviour of reactive systems but at a low abstraction level
    - Higher-level modelling abstractions that maps the ones used in the design (Components, Nodes and Networks) are needed
      - They exist in MARTE, but a formalization of their mapping to the SAM chapter should be defined.
      - ¿Only MARTE? ¿SysML? ¿UML 2?

- Analysis tools
  - Non-linear transactions are very typical when combining different components => We have used simulation, but analytical tools must be implemented