Real-time Extensions to the OMG's "Deployment and Configuration of Component-based Distributed Applications" Specification

Patricia López, José M. Drake, and Julio L. Medina

Grupo de Computadores y Tiempo Real
Universidad de Cantabria, Spain

OMG's Ninth Workshop on Distributed Object Computing for Real-time and Embedded Systems
July 14-16, 2008 - Arlington, VA USA

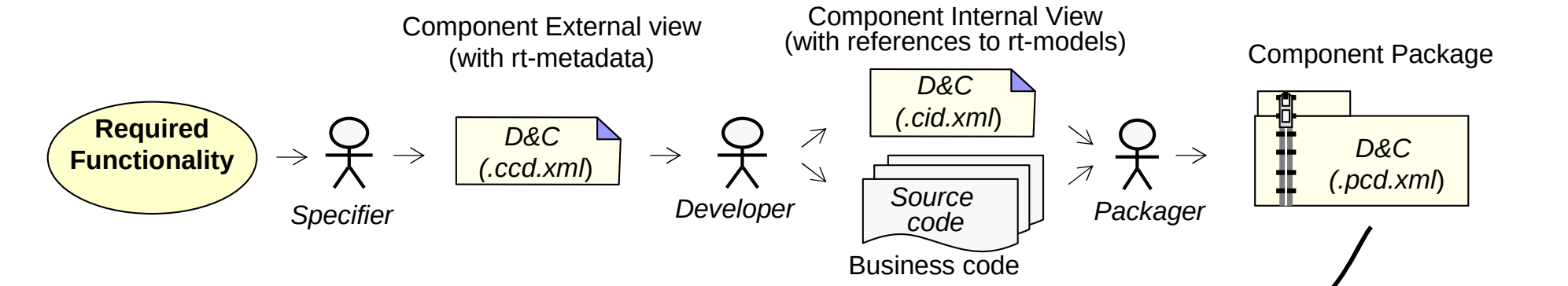
Funded by the European Union under contracts FP6/2005/IST/5-034026, IST-004527, and FP7- 224330 and by the Spanish Government under grant TIC2005-08665-C03

Introduction

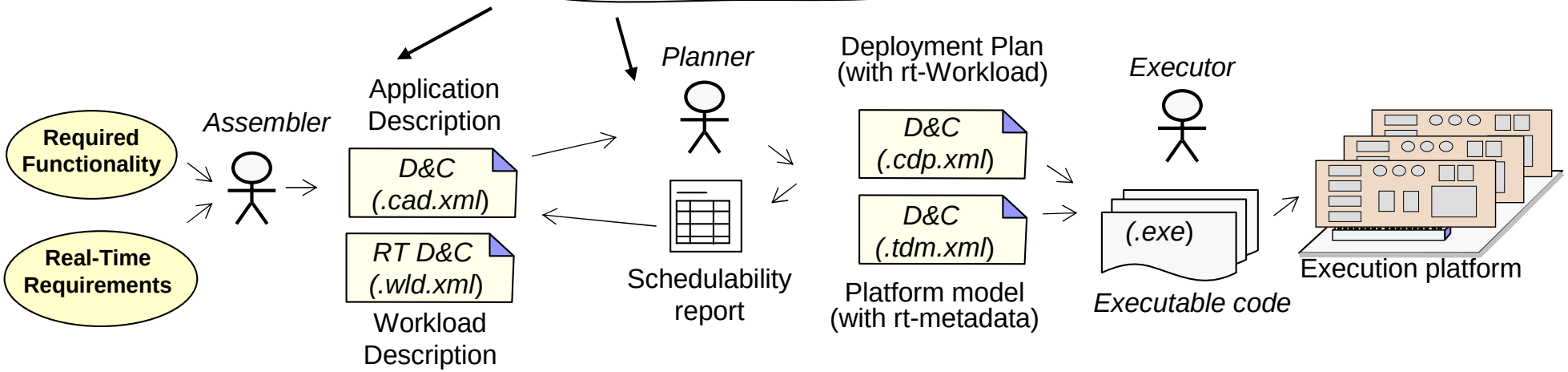
- Real-Time => Timing predictability, schedulability.
- Static schedulability analysis based on RMA techniques is made using scenario models.
- Component-based strategies simplify the process to get the analysis models of real-time applications:
  - Component packages must include the information about the temporal behavior of the component code as non-functional metadata.
  - In the application design phase, a tool, driven by the deployment plan, composes the real-time data of its constituent components and built the complete real-time model of the application.

Real-Time aspects in component-based development

Component design



Application design



Our proposal

- The proposal is to promote a component based strategy and extend the D&C specification to include the metadata and the tasks required to manage the real-time models along the envisioned development process.

Our proposal in brief (1)

1. Extend the D&C specification to include:
   - In Component Interface metadata:
     - Conditions for connections so that the assembled set has a predictable timing behavior.
     - For active components, the description of the end-to-end flow transactions that may be started on it.
   - In Component Implementation metadata:
     - References to the models that describe the RT behavior of each of its implementations.
   - In Application Assembly metadata:
     - The analysis contexts in which the application is to be scheduled: workload and time constraints.
   - In Target Data Model:
     - References to the models that describe the processing capacity of the platform resources.
   - In Deployment Plan:
     - The assignment of values to those configuration parameters related to scheduling (priorities, deadlines, resource reservation contracts, etc.)

Our proposal in brief (2)

2. Limit information and concerns so that:

- The designers of the application (assembler and planner):
  - Do not need to know the real-time modeling methodology.
  - Use tools that analyze the temporal behavior of the application.
  - Interpret the results of the tools as references to a conceptual frame defined in the D&C rt-extensions.

- The developer of the components, and the platform, who know their code, have to master the RT modeling methodology used to construct the corresponding real-time models.

Our proposal in brief (3)

3. Extend the deployment process so that it includes the tasks that are inherent to the real-time applications design

   - In the design phase it is evaluated whether with the available components the application may have a complete real-time model or not.

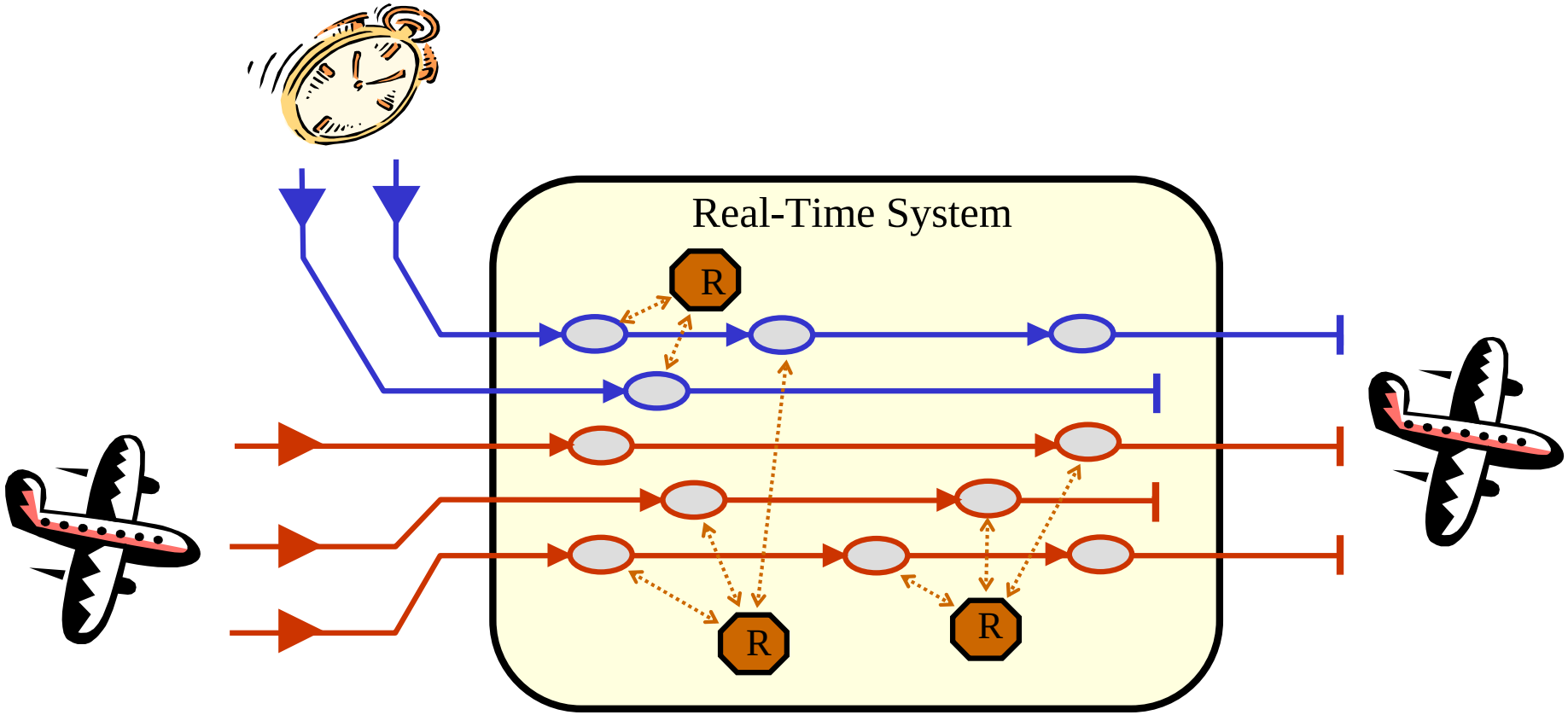
   - In the application development phase the real-time model is assembled and used to:

     - Evaluate schedulability

     - Calculate application's configuration parameters like, allocation, priorities, or the contracts to negotiate with the managing services in the deployment over a contract-based platform.

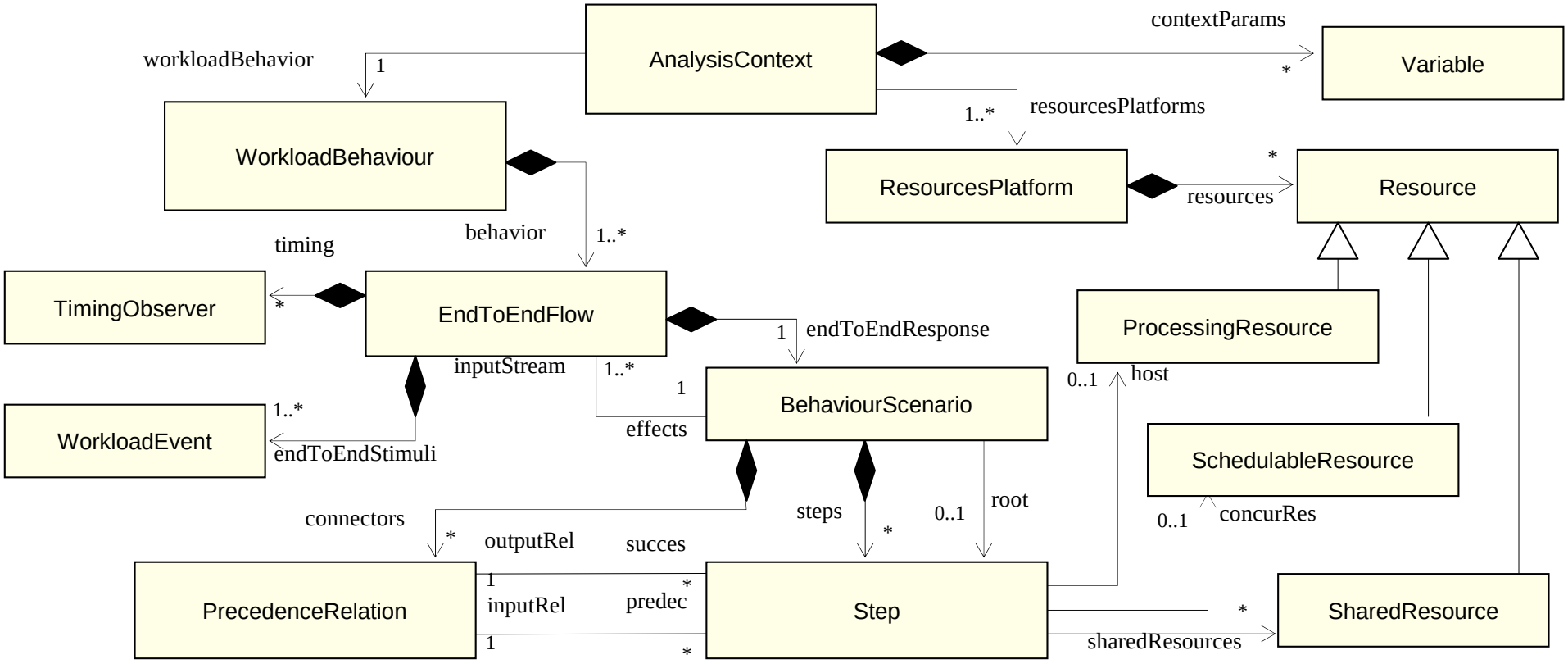
Reactive model of an application

- The real-time analysis models are described as reactive scenarios running over a scheduled platform.



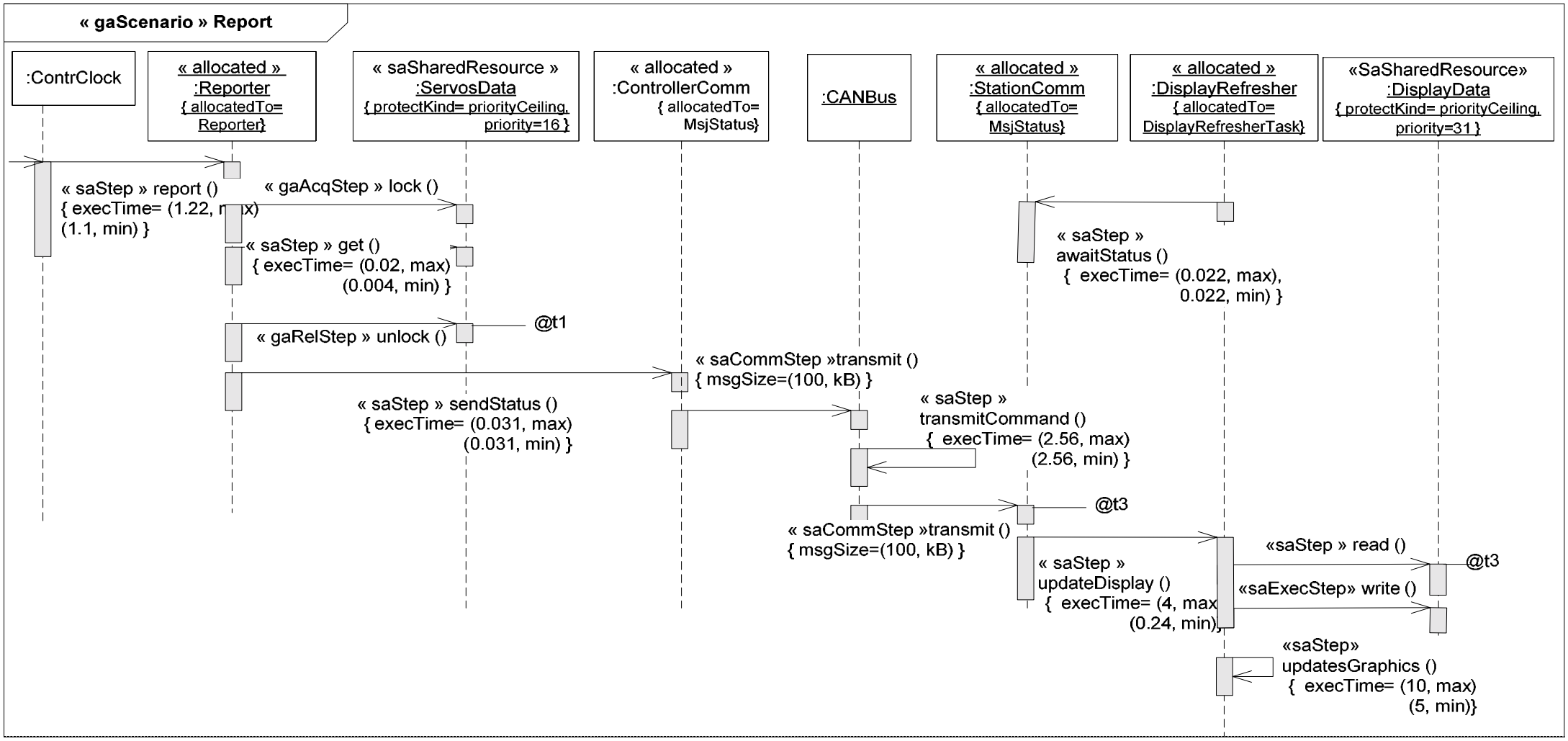
Reactive model in MARTE

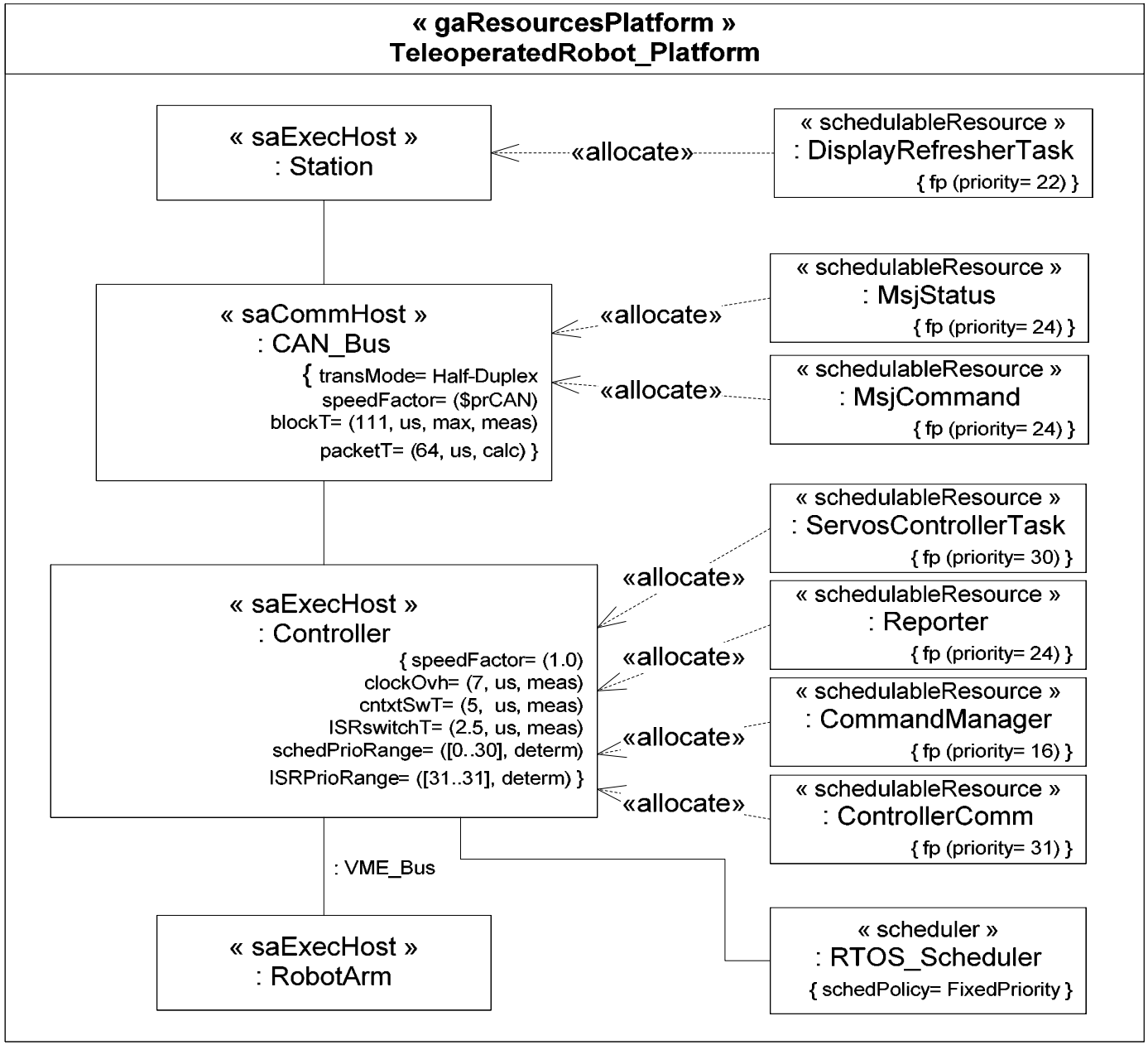
- These modeling capabilities are provided in the Analysis sub-profiles of the UML profile for MARTE



P. López, J. Drake, and J. Medina
Grupo de Computadores y Tiempo Real - Universidad de Cantabria

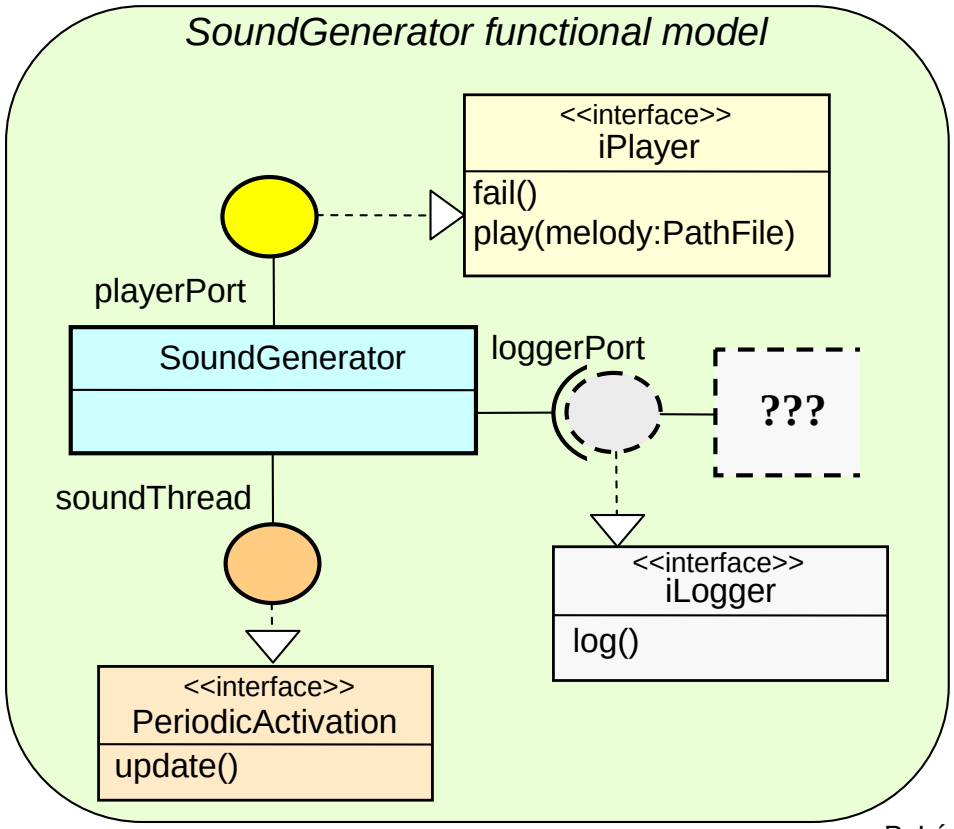
Reactive model example

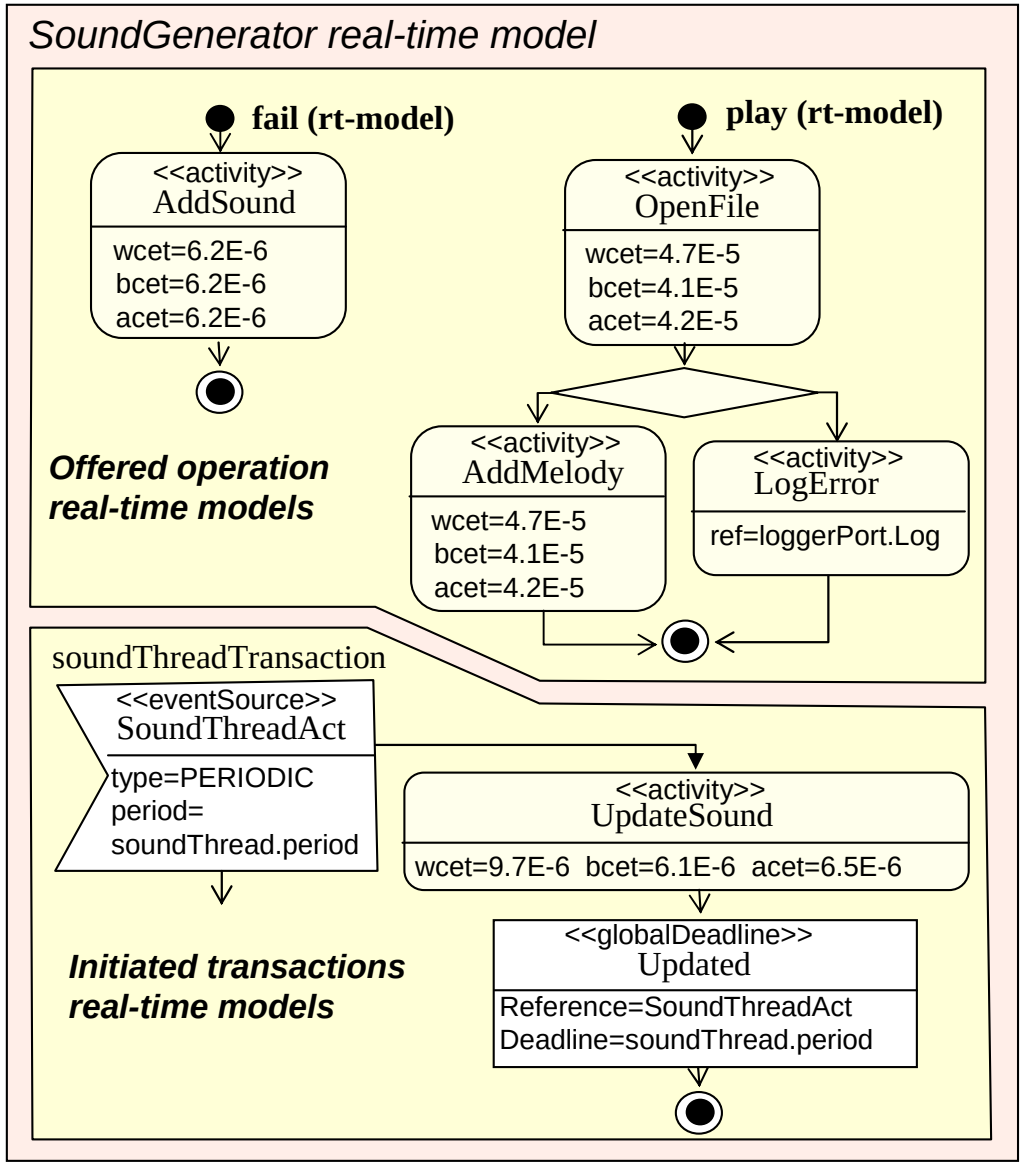



« gaResourcesPlatform »
TeleoperatedRobot_Platform
« saExecHost »
: Station
«allocate»
« schedulableResource »
: DisplayRefresherTask
{ fp (priority= 22) }
« saCommHost »
: CAN_Bus
{ transMode= Half-Duplex
speedFactor= ($prCAN)
blockT= (111, us, max, meas)
packetT= (64, us, calc) }
«allocate»
« schedulableResource »
: MsjStatus
{ fp (priority= 24) }
«allocate»
« schedulableResource »
: MsjCommand
{ fp (priority= 24) }
« saExecHost »
: Controller
{ speedFactor= (1.0)
clockOvh= (7, us, meas)
cntxtSwT= (5, us, meas)
ISRswitchT= (2.5, us, meas)
schedPrioRange= ([0..30], determ)
ISRPrioRange= ([31..31], determ) }
«allocate»
« schedulableResource »
: ServosControllerTask
{ fp (priority= 30) }
«allocate»
« schedulableResource »
: Reporter
{ fp (priority= 24) }
«allocate»
« schedulableResource »
: CommandManager
{ fp (priority= 16) }
«allocate»
« schedulableResource »
: ControllerComm
{ fp (priority= 31) }
: VME_Bus
« saExecHost »
: RobotArm
« scheduler »
: RTOS_Scheduler
{ schedPolicy= FixedPriority }

Real-time model of a component

- The real time model of a component includes all the information related to its internal code that is required to predict the temporal behavior of any application in which the component may be used.



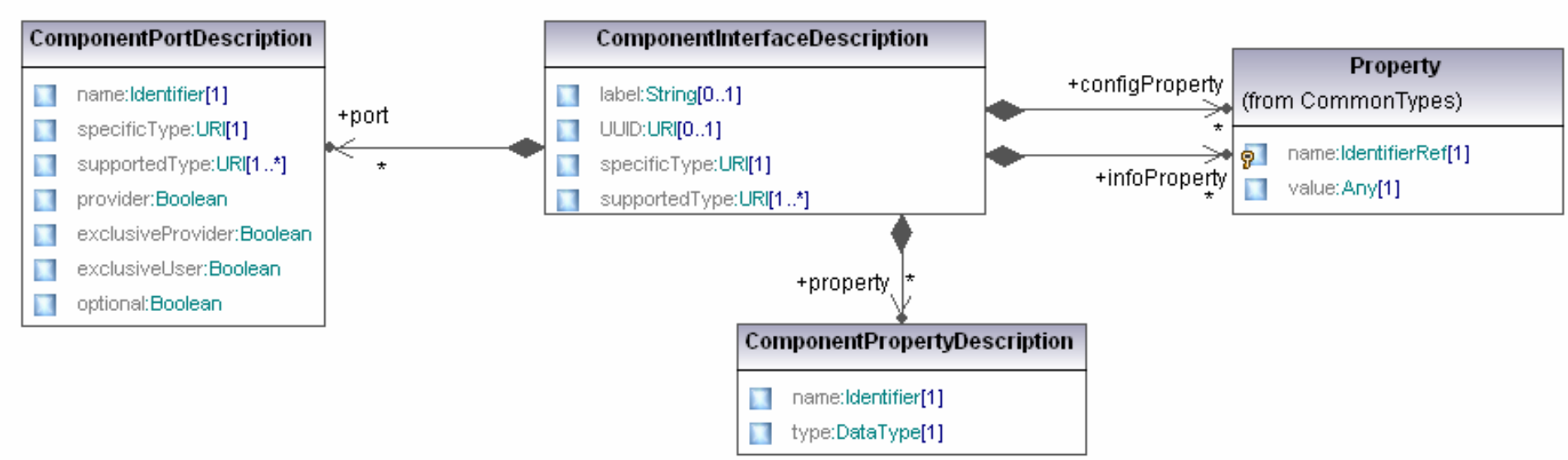


Real-time Models in the D&C specification

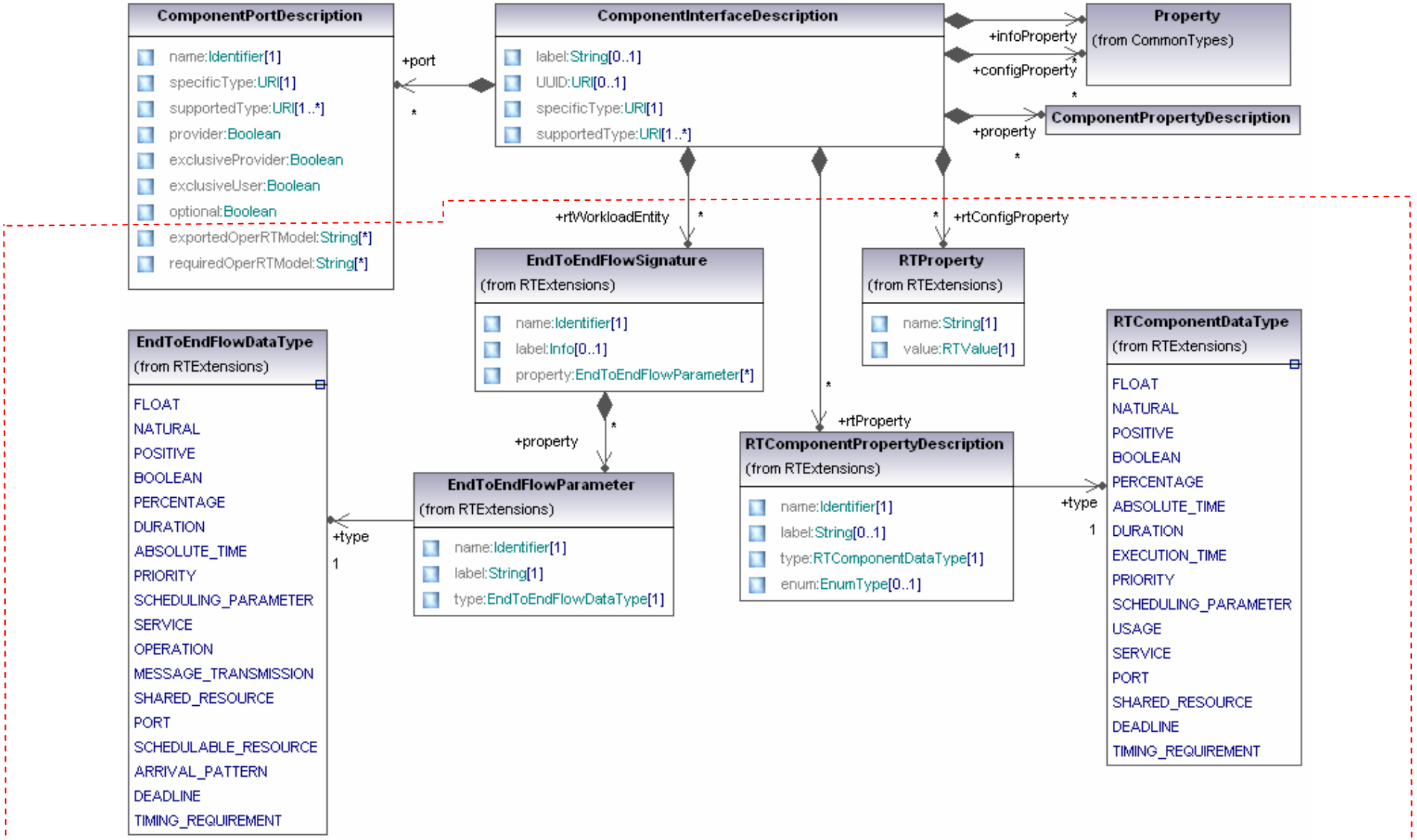
- The developer is who elaborates the real-time model of a component.
  - A concrete modeling methodology that allows for the composition of models must be used
  - We propose to use CBS MAST, an extension of MAST
  - The models are included as metadata associated to the component's implementation by means of the *Component Implementation Description*
- The assembler and the planner do not need to access the internal representation of the real-time models, they require only the information that is necessary to decide if the utilization of one particular component may lead to a complete real-time model, and hence that the predictability of the application may be evaluated
  - They do not need to master any real-time modeling methodology
  - The required information is included as metadata in the external description of the component, this is in the *Component Interface Description*.

D&C's Component Interface Description

- It represents the external description of the component. It is used to:
  - Decide the utility of a component in an application
  - Know the connectivity and configuration characteristics of the component
- It is defined through:
  - The set of ports the component offers
  - The set of ports the component requires
  - The configuration properties that it admits



Extended D&C's Component Interface Description

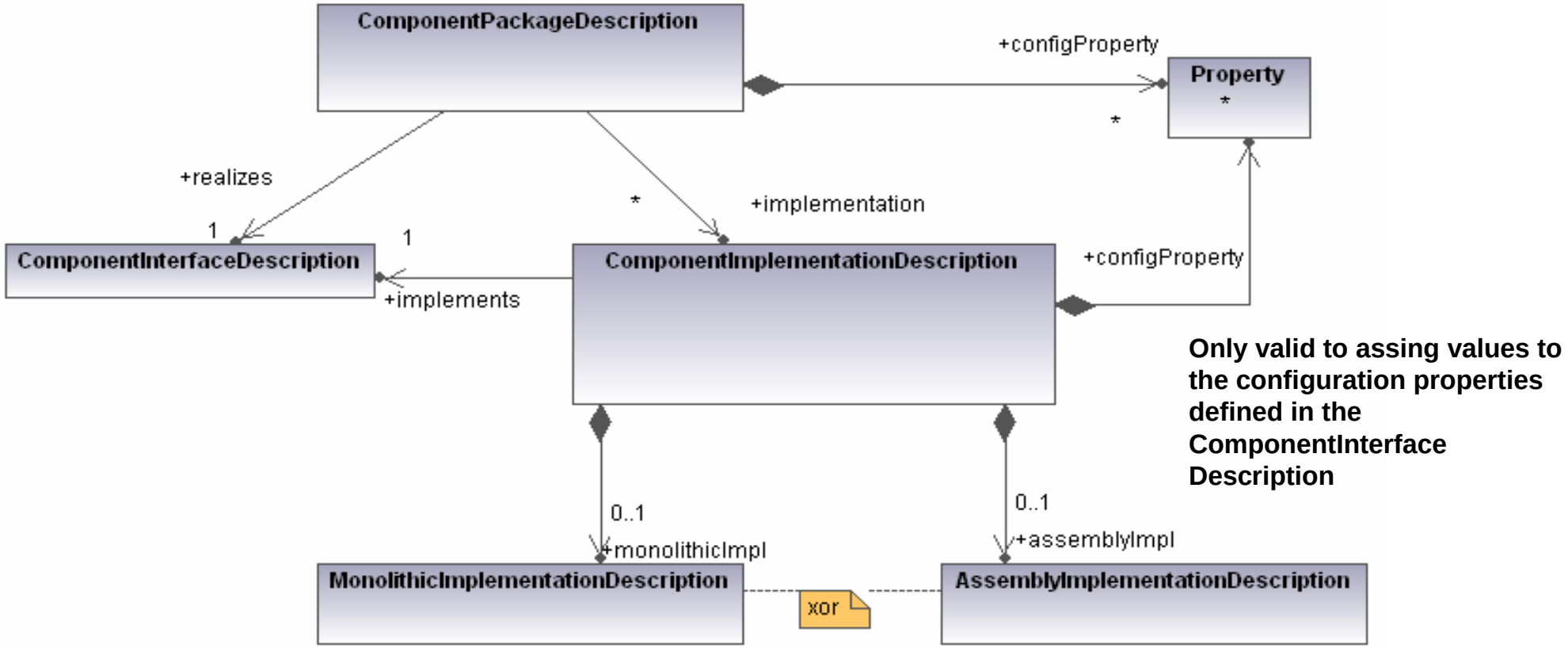


Example of Component Interface Description

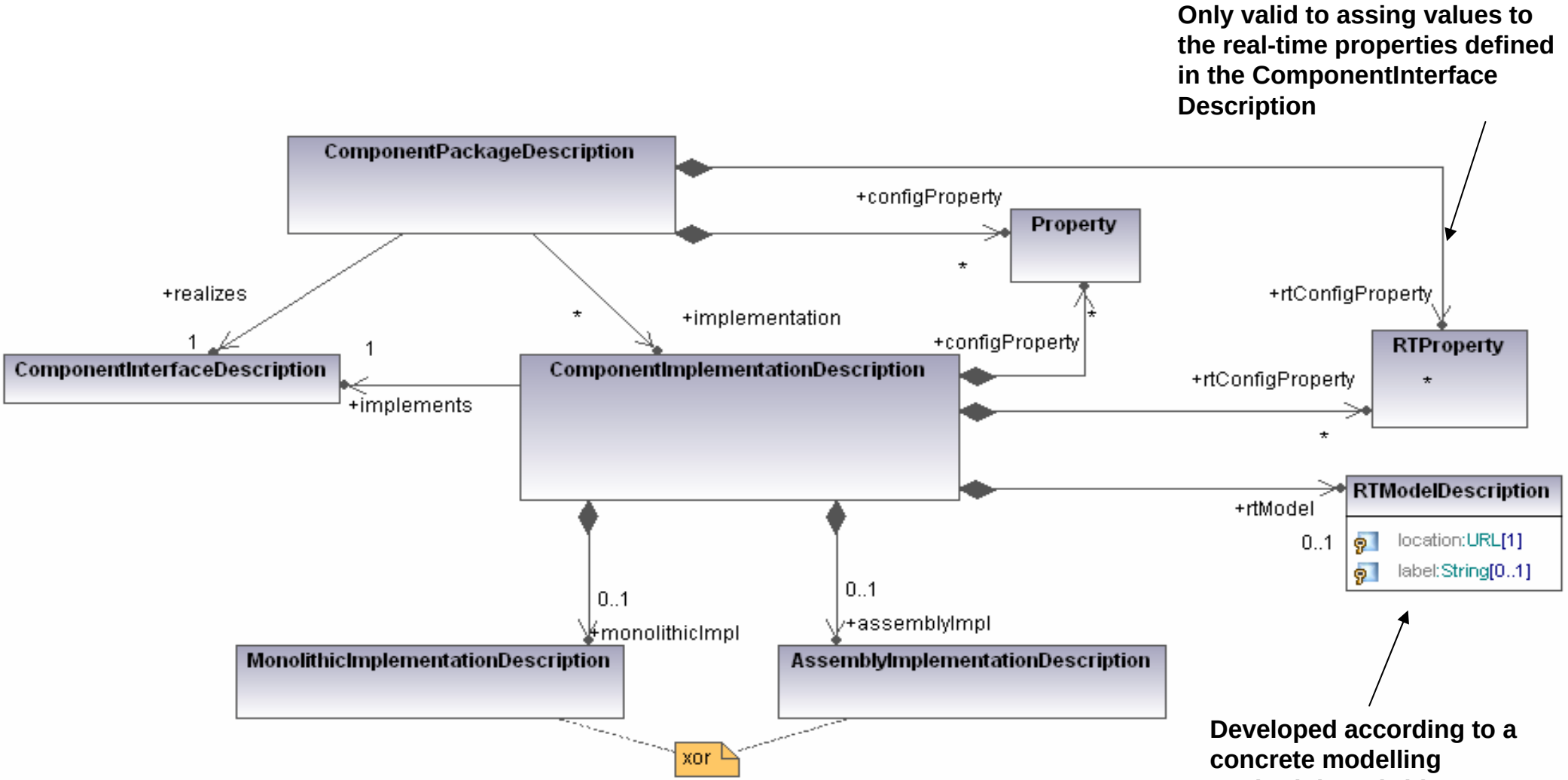
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<DnCcdm:componentInterfaceDescription xmlns:xmi="http://www.omg.org/XMI"
    …
    <description label="AdaMaRTE SoundGenerator  Service"
     specificType="components/multimedia/SoundGenerator.ccd.xml">
---
<!-- *** FACETS DECLARATION *** -->
      <port name="playerPort"
           specificType=interfaces/multimedia/iPlayer.idl.xml::multimedia::iPlayer
           provider="true"
           exclusiveUser="true"
           kind="FACET"
           exportedOperRTModel="play playMany fail"/>

<!-- *** RECEPTACLES DECLARATION *** -->
      <port name="loggerPort" specificType=interfaces/database/iLogger.idl.xml::database::iLogger
           provider="true"
           exclusiveUser="true"
           kind="RECEPTACLE"
           requiredOperRTModel="log"/>
…

<!-- ***ATTRIBUTES DECLARATION***-->
    <property name="mode" type=interfaces/multimedia/iPlayer.idl.xml::multimedia::PlayingMode/>


<!--***TRANSACTIONS DECLARATION***-->
      <rtWorkloadEntity name="soundThreadTransaction" label="…">
            <transactionProperty name="period" type="DURATION"/>
      </rtWorkloadEntity>
</ComponentInterfaceDescription>
```

D&C's Component Package Description

- It describes multiple implementations of the same component interface
- Each implementation can be monolithic or assembly based (in that case it is described as the set of instances and connections that form it).



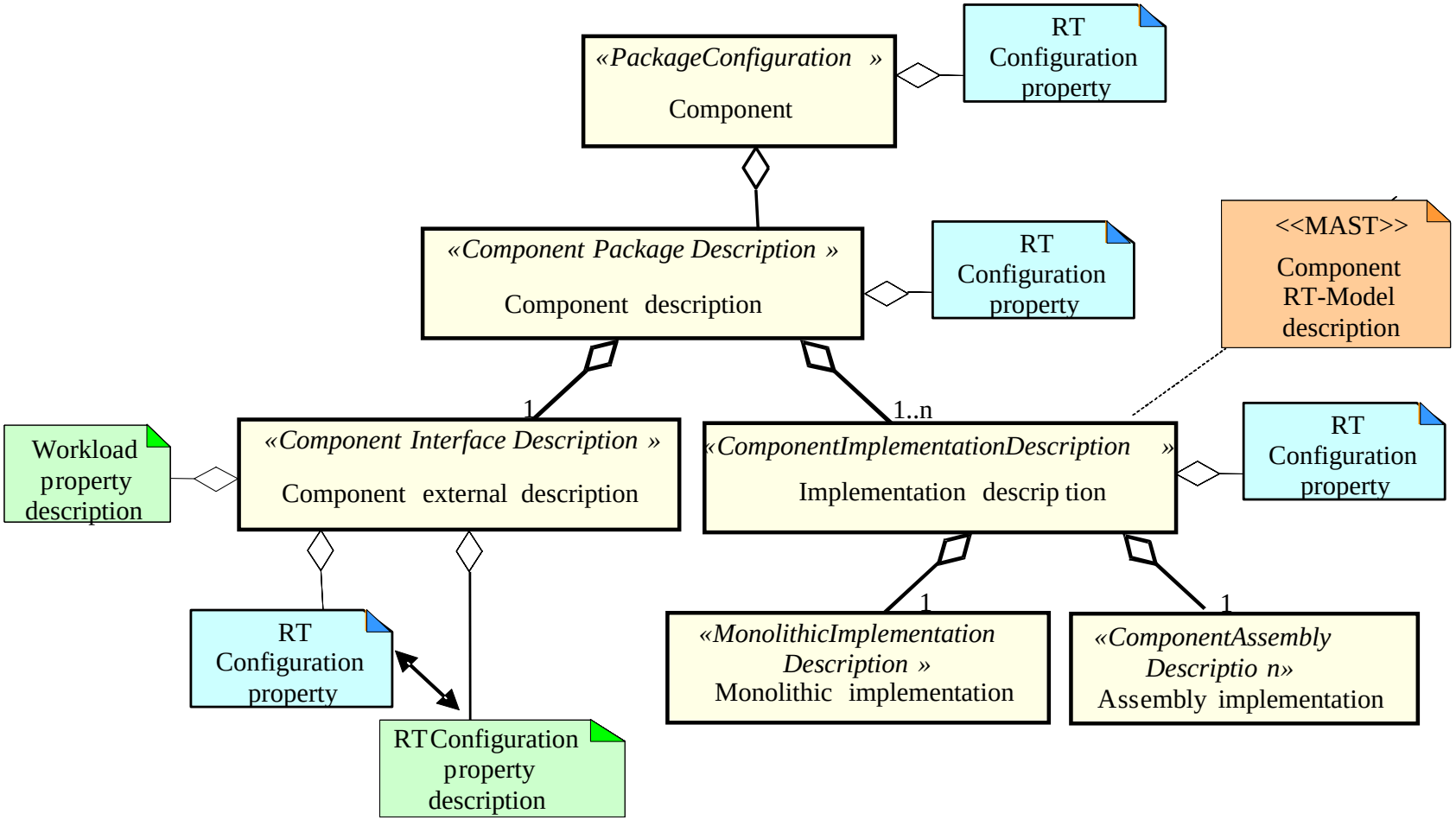
Extended D&C's Component Package Description



Example of Component Package Configuration

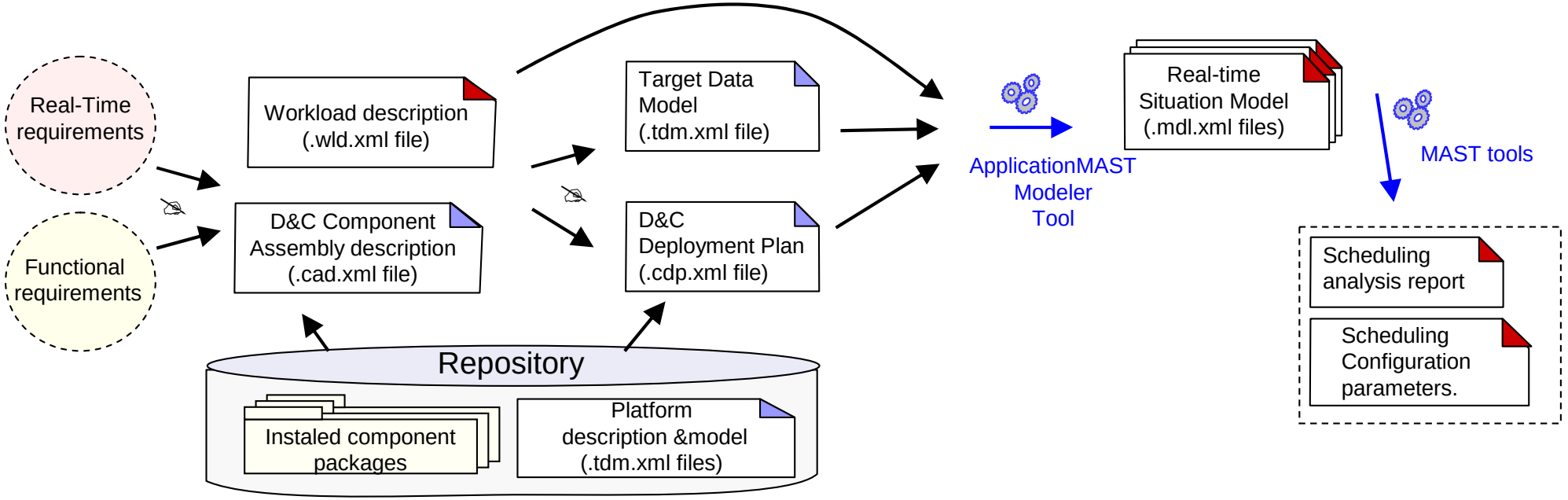
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<DnCcdm:packageConfiguration xmlns:xmi="http://www.omg.org/XMI"

  <basePackage>
    <realizes><ref>component/multimedia/SoundGenerator.ccd.xml</ref></realizes>
   <implementation name="MaRTE_SoundGenerator rtModel="component/multimedia/SoundGenerator.rtm.xml>
        …
        <monolithicImpl>
     <primaryArtifact name="MaRTE_SoundGenerator.adb">
        <description location="component/multimedia/soundGenerator/MaRTESoundGenerator.adb"/>
            </primaryArtifact>
             <primaryArtifact name="MaRTE_SoundGenerator.ads">
        <description location="component/multimedia/soundGenerator/MaRTESoundGenerator.ads"/>
            </primaryArtifact>
     <primaryArtifact name="SoundGenerator_business_interface.adb">
        <description
              location="component/multimedia/soundGenerator/SoundGenerator_Business_Interface.adb"/>
            </primaryArtifact>
            <deployRequirement resourceType = "OS" name="OS_Requirement">
        <property name="type"><value>MaRTE_OS</value></property>
     </deployRequirement>
     </implementation>
  </basePackage>
</packageConfiguration>
```

Summary of RT Components Modeling



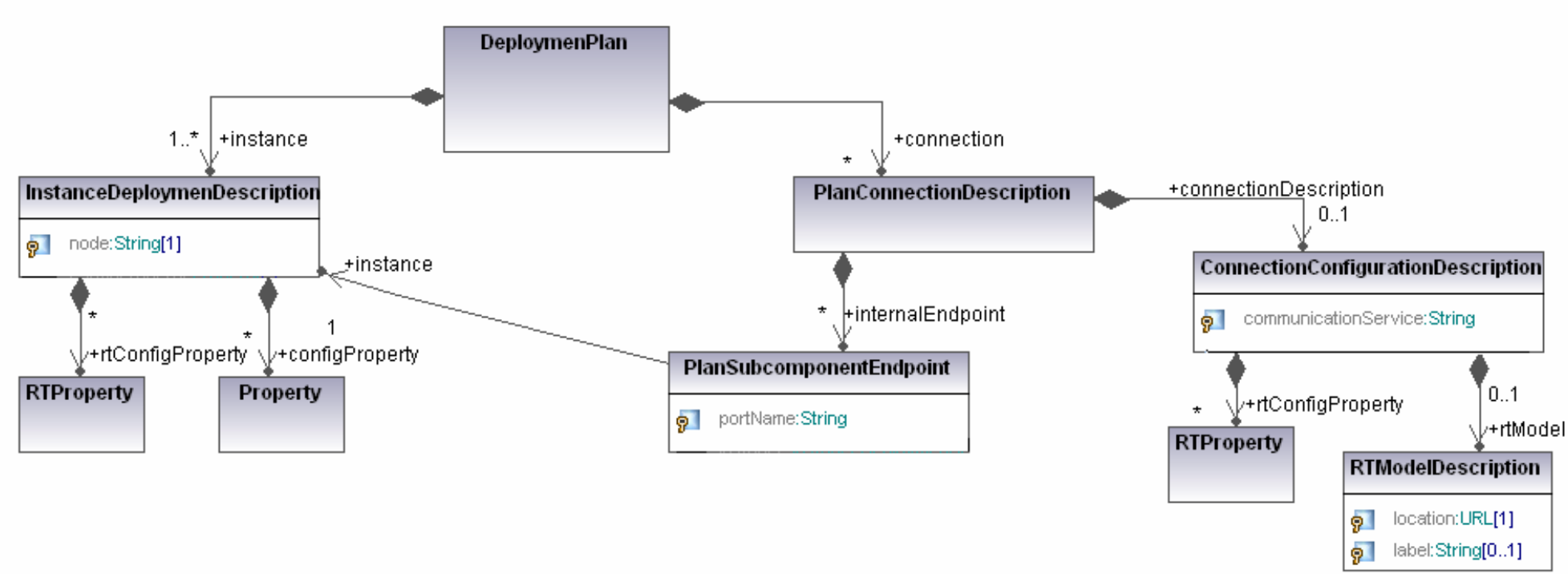
Real-time model of an application: AnalysisContext

- AnalysisContext => a concrete mode of operation of a system to be analyzed. It is generated from:
  - Platform: It is defined according to the extended D&C's domain description
  - Deployment Plan: instances, their connections, the assignment of instances to the nodes, and the communication mechanisms.
  - Workload: Stimuling events, Extension to D&C



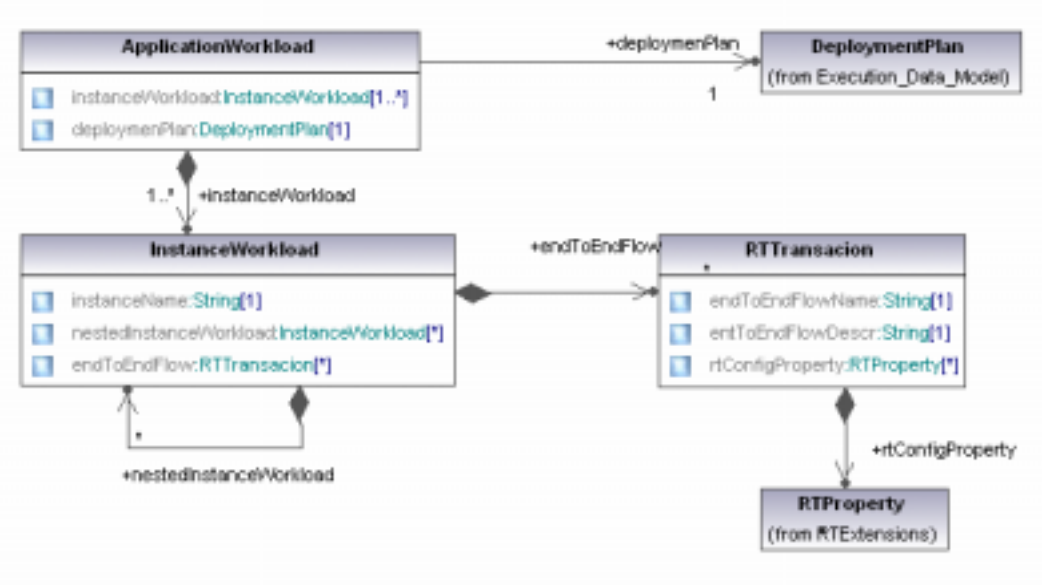
Extended D&C's Deployment Plan

- This model is enriched with the real-time extension for the assignment of scheduling configuration parameters, applied both to component instances and connections:
  - For the instances: those corresponding to the real-time configuration properties defined in the component interface description. Ex: threads priorities or deadlines, ceiling priorities or synchronization artifacts, etc.
  - For the communication mechanisms: messages priorities or deadlines, priority of the threads that perform the message dispatching, etc.



Workload Model

- Workload Model: New in D&C, it is expressed as a set of transactions
- All the transactions associated to any of the instances of a component in the deployment plan must be declared.
- They are parameterized to be adapted for each usage of the component,
  - The parameters are those declared in the Component Interface Description.
  - The workload model assigns values to them.



Example of application workload

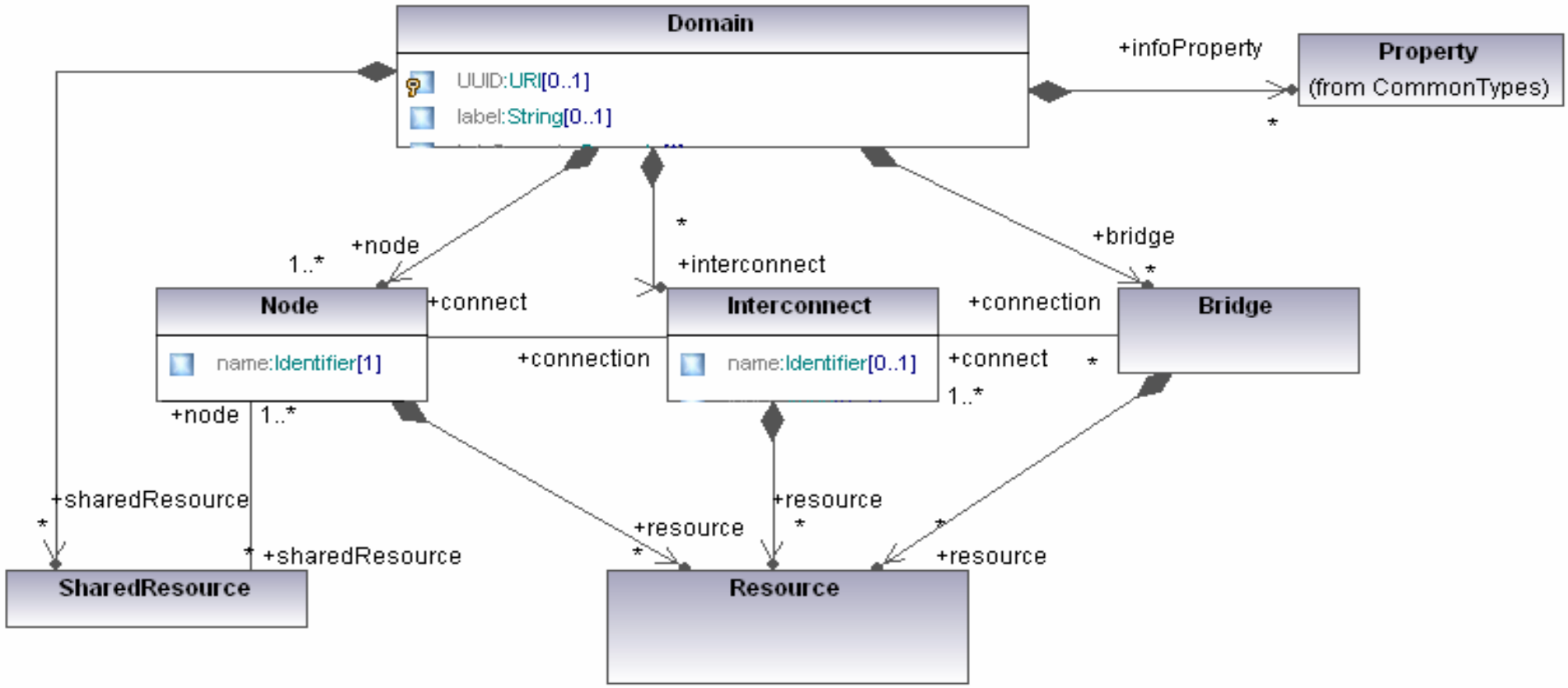
```
<?xml version="1.0" encoding="UTF-8"?>
<rtWkld:applicationWorkload xmlns:rtWkld="http://ctr.unican.es/cbsdnc/rtWorkload"
    xmlns:DnCbt="http://ctr.unican.es/cbsdnc/DnC_CCM_BasicTypes"
    …
    deployment_plan="scs/applications/scs/jetFollower_deployment.xml">
    <instanceWorkload instancename="alarmSound">
        <endToEndFlow endToEndFlowDescr="soundThreadTransaction"
    endToEndFlowName="theSoundTrans">
            <rtConfigProperty name="period">
                <value>
                    <duration>1.0</duration>
                </value>
            </rtConfigProperty>
        </endToEndFlow>
    </instanceWorkload>
</rtWkld:applicationWorkload>
```

Real-time model of the platform resources

- To analyze the system it is necessary to have also the real-time model of the platforms that will be used.
- D&C does not mention the necessity of storing descriptors of platform models => A extension is required to support the handling of parameterized platform models
- Elements in the platform model are: Processing resources (processors or networks), schedulers, scheduling policies, threads, control access protocols.
- The processing capacity of the processors and networks is expressed as a speed factor. This is used in combination with the Normalized Execution Times expressed in the description of the operations to get the actual execution time in a platform.

D&C's Target Data Model

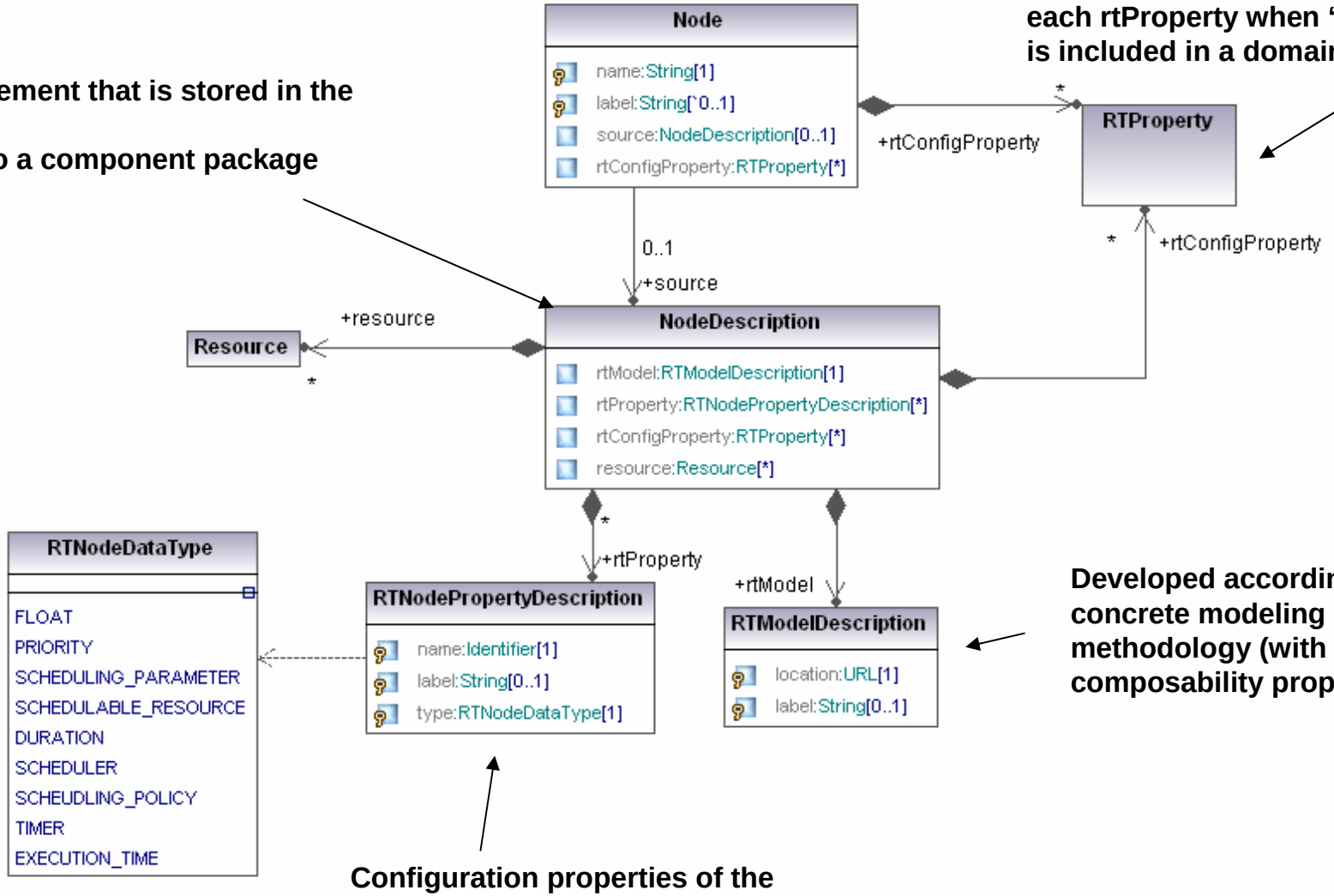
- It describes the concrete platform in which the component-based application is going to be executed



Model of a Node

This is the element that is stored in the repository (in analogy to a component package description)

Concrete values can be assigned to each rtProperty when "node instance" is included in a domain



Developed according to a concrete modeling methodology (with composability properties)

Configuration properties of the real-time model

Example of NodeDescription and Node

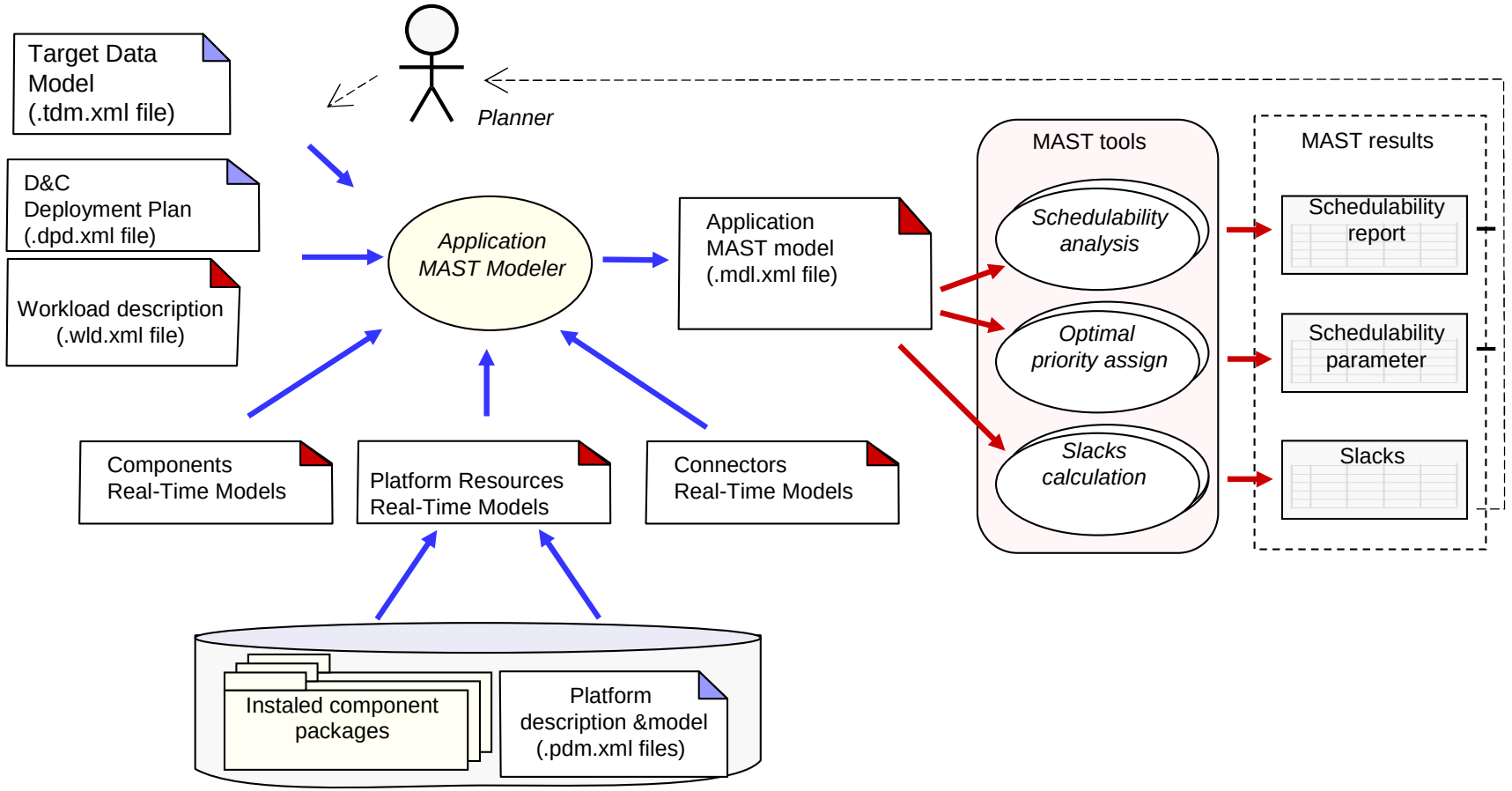
```
<?xml version="1.0" encoding="UTF-8"?>
<DnCtdm:nodeDescription xmlns:DnCtdm="http://ctr.unican.es/cbsdnc/DnC_CCM_TargetDataModel"
    :DnCbt="http://ctr.unican.es/cbsdnc/DnC_CCM_BasicTypes" :DnCct=http://ctr.unican.es/cbsdnc/DnC_CCM_CommonTypes
    label="Description of MaRTEOS node">
    <resource resourceType="OS" name="theOS">
        <property name="type" kind="ATTRIBUTE">
            <value>MaRTEOS</value>
        </property>
    </resource>
    <rtModel location="scs/platform/gral/MaRTEOS_2_2.rtm.xml"/>
    <rtProperty type="FLOAT" name="speed_factor"/>
    <rtConfigProperty name="speed_factor">
        <value>
            <float>1.0</float>
        </value>
    </rtConfigProperty>
</DnCtdm:nodeDescription>
```

MaRTE_OS_2_2.cnd.xml

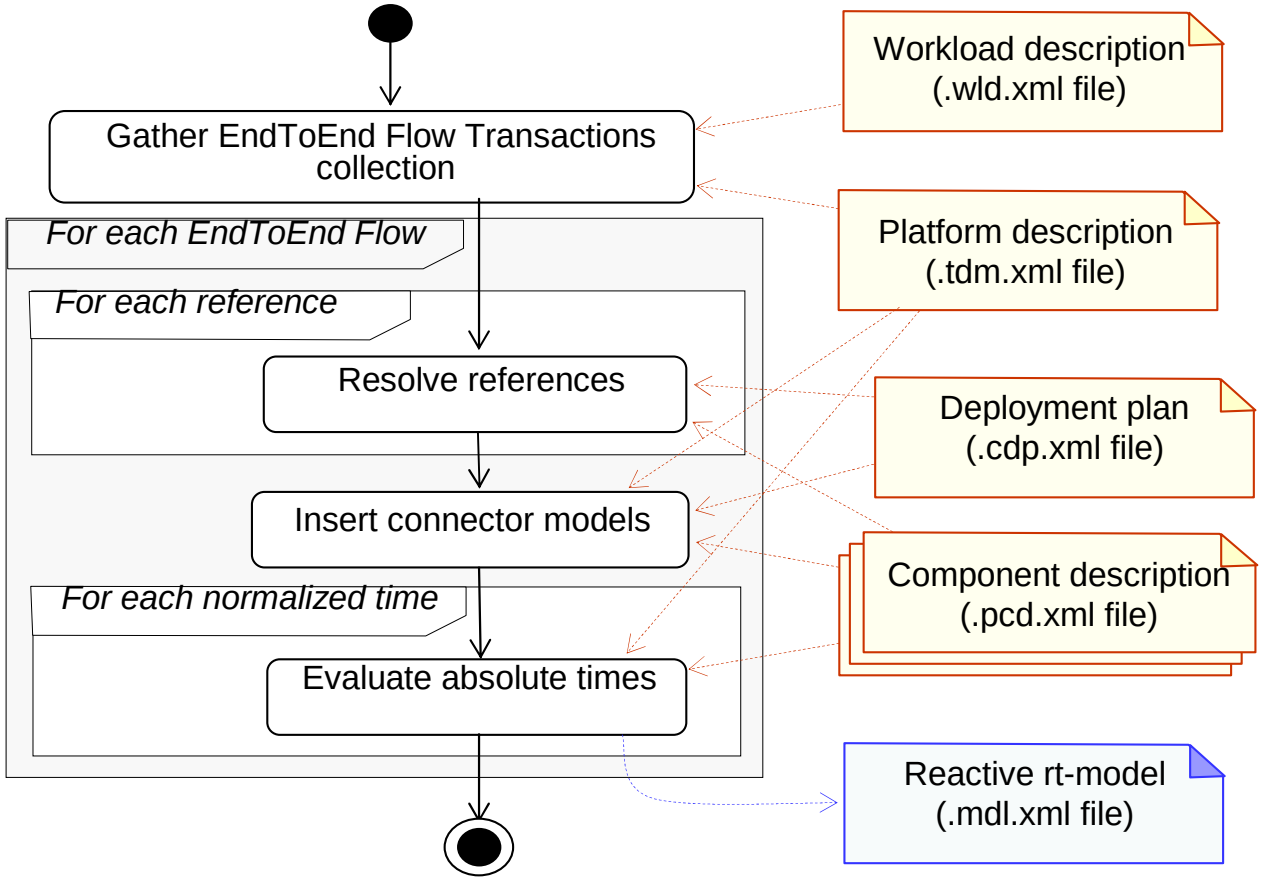
```
<DnCtdm:domain xmlns:DnCtdm="http://mast.unican.es/cbsdnc/DnCTargetDataModel"
    …
    <node name="central" label="Central node" source="scs/platform/gral/MaRTEOS_2_2">
        <rtConfigProperty name="speed_factor">
            <value>
                <float>0.8</float>
            </value>
        </rtConfigProperty>
        <connection>theNetwork</connection>
    </node>
    …
</DnCtdm:domain>
```

aDomain.tdm.xml

Composition of real-time models



Herramienta de composición de modelos



Conclusions

- We have proposed a number of modeling practices and a methodology that extends the D&C specification to include the metadata and the tasks required to manage the real-time models along a component-based development process.

- This extension allows the designers of real-time component-based applications to build their models and then analyze them using only the set of basic concepts included in the RT-D&C extension, without requiring expertise in the real-time modeling methodology used by the developers of the components to formulate their respective analysis models.

- The experiments made to validate this approach, have lead to a successful component-based development suite using Ada [*]

[*] P. López, J.M. Drake, P. Pacheco, and J.L. Medina, An Ada 2005 Technology for Distributed and Real-Time Component-based Applications, in Proc. of the *13th Intl. Conference on Reliable Software Technologies Ada-Europe*, Venice, 2008