

# UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) Tutorial

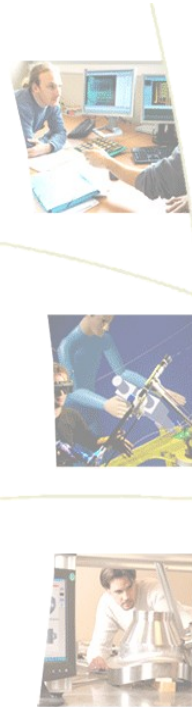
## Software Resource Modeling

Workshop on Distributed Object Computing for Real-time and Embedded Systems

Washington, DC, USA  
July 14th, 2008

Frédéric Thomas, Sébastien Gérard, [Chokri Mraidha](#)

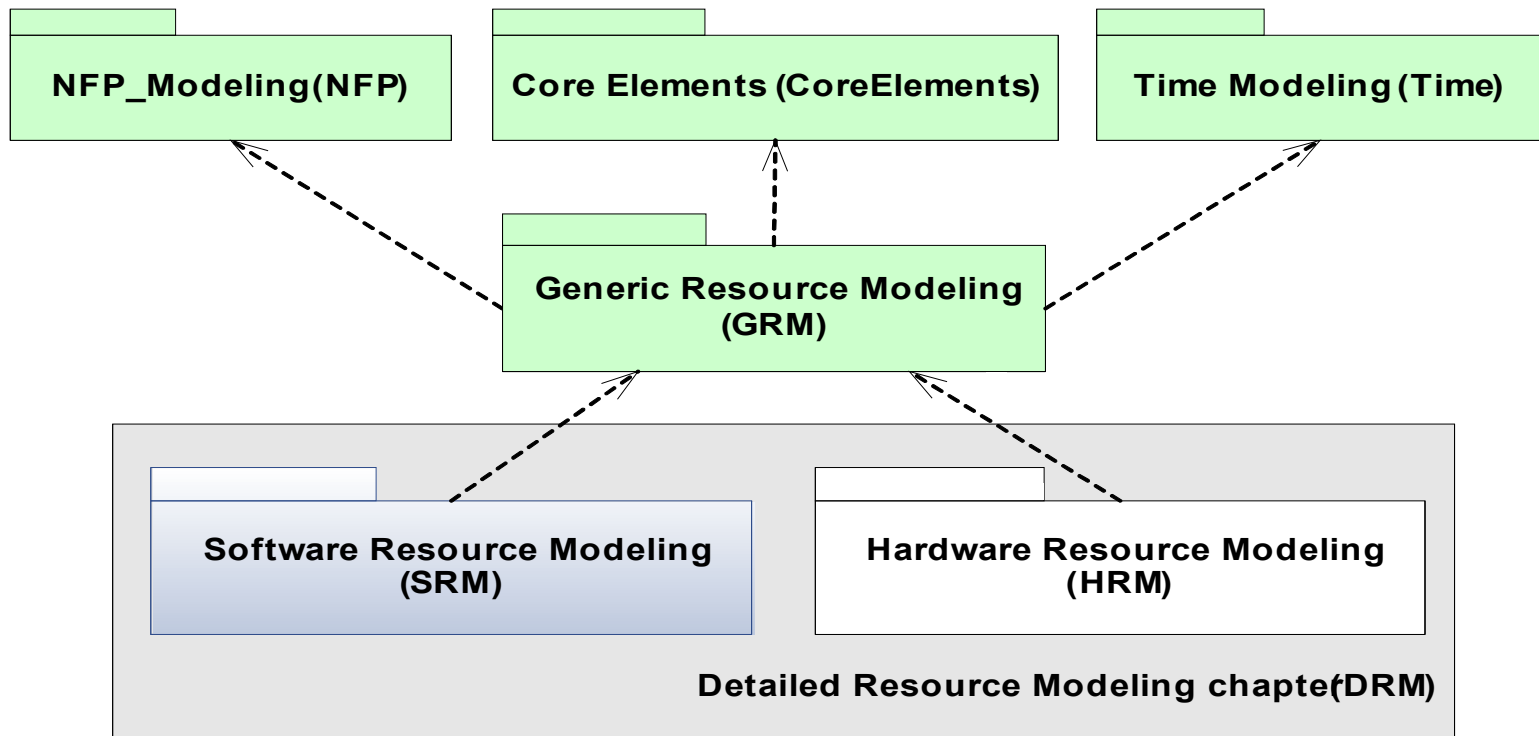
[Chokri.Mraidha \[at \] cea.fr](mailto:Chokri.Mraidha[at]cea.fr)



# Outline

- SRM Overview
  - What is the SRM profile?
  - In which design steps shall I use SRM?
  - In which typical cases shall I use SRM?
- RTOS API modeling with SRM: the OSEK/VDX case study
  - Why shall I use SRM for RTOS API modeling?
  - OSEK/VDX overview
  - What is supported by the SRM profile?
  - The OSEK/VDX Task modeling with SRM
  - The OSEK/VDX Event modeling with SRM
- Examples of API model uses
  - A robotic case study
    - multitask model designs
    - OS configuration file generation
    - RTE application models porting

# SRM overview



# What is the SRM Profile ?

- **The Software Resource Modeling profile is:**

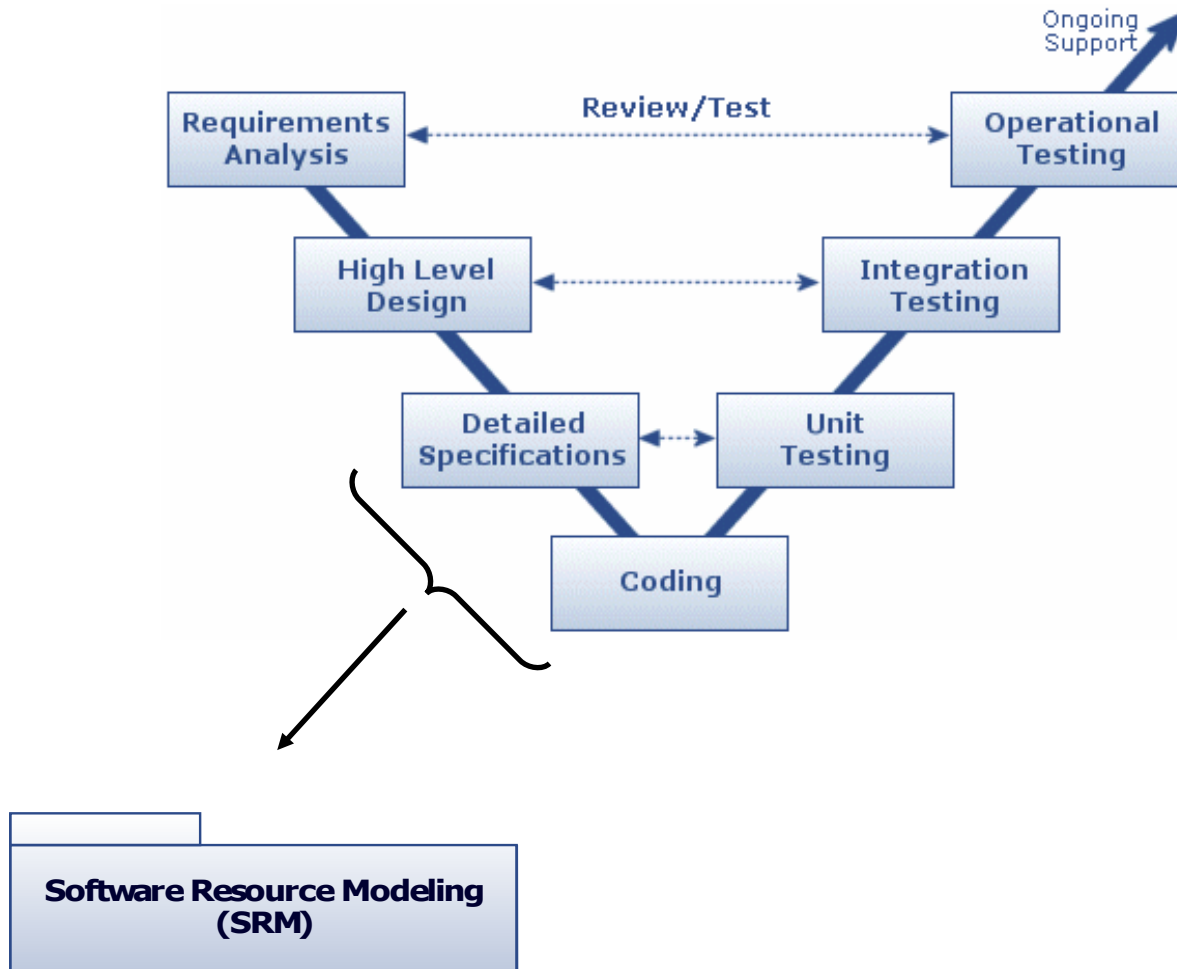
- **A UML Profile to describe API of software execution supports**

- Real Time Operating Systems (RTOS)
- Language Libraries (e.g. ADA)

- **BUT, the SRM profile is not a new API standard dedicated to the Real-Time and Embedded domain.**

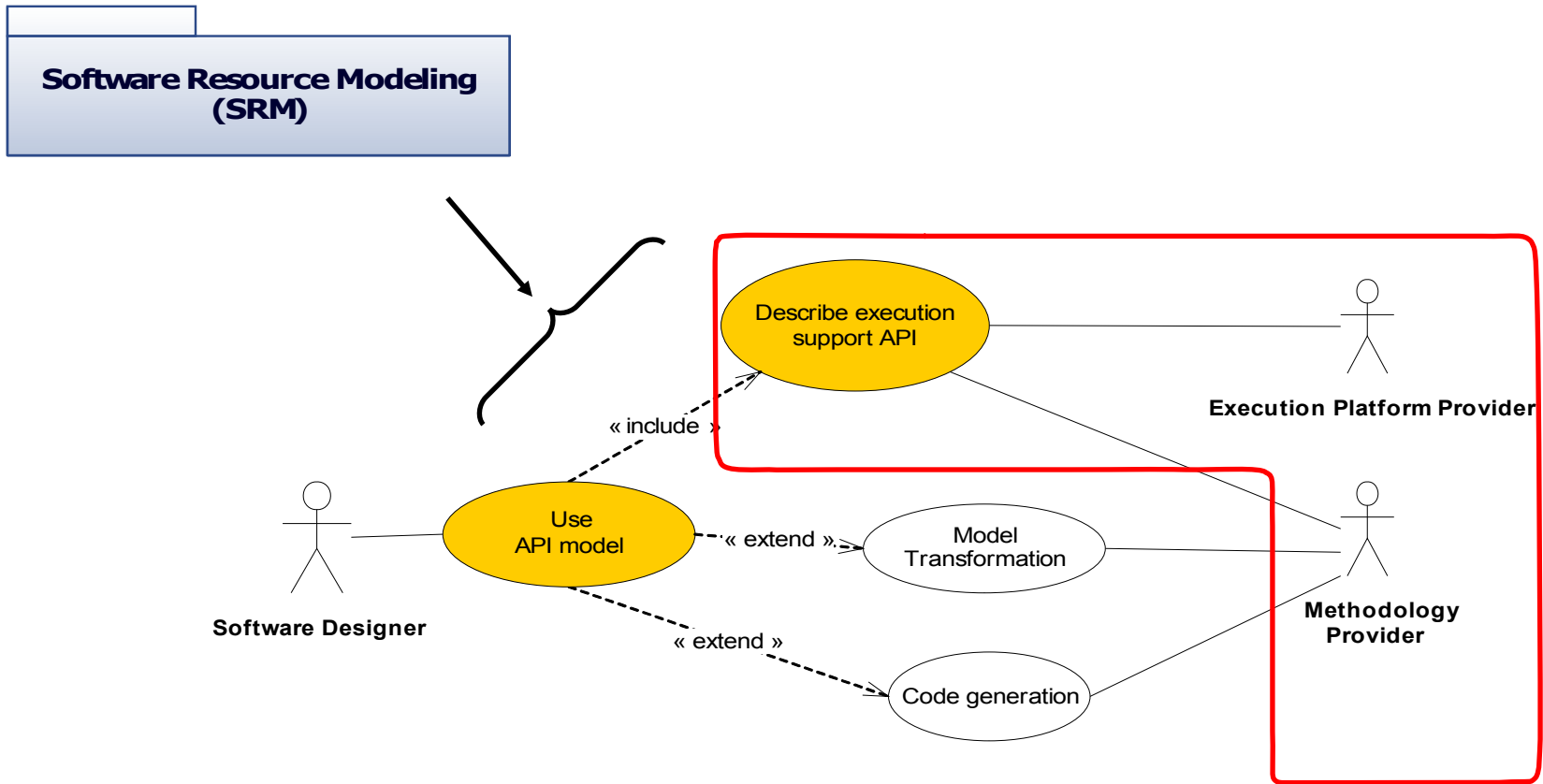
- SRM allow users to describe RTE API involved in the design cycle
  - standard RTOS API (e.g. POSIX, OSEK/VDX and ARINC 653)

# In which steps shall I use SRM ?



**Software Resource Modeling (SRM)**

# In which typical cases shall I use SRM ?



# Why shall I use SRM for RTOS API modeling ?

## ● RTOS API modeling with UML is already possible

– But,

- UML core is lacking in some key RTE native artifacts
  - RTOS providers have no modeling artifacts to describe tasks, semaphores, mailboxes ...
  - Methodology providers can't describe generic tools
    - » For each model they must describe specific generative tools (e.g. code generator, model transformations)

## ● SRM profile allows

- To describe efficient and precise multitask models
- To be able to describe generic generative tools
- To describe models in an unified and standard way
  - SRM profile is a sub-profile of the MARTE standard

list

cei



# Execution support API modeling : the OSEK/VDX case study

## ● Let's take an example :

– OSEK/VDX standard (<http://www.osek-vdx.org>)

- It aims to provide to the automotive industry a standard for an open-ended architecture for distributed control units in vehicles
- The open architecture introduced by OSEK/VDX comprises these three main areas:
  - OSEK COM : Communication (data exchange within and between control units)
  - OSEK NM : Network Management (Configuration determination and monitoring)
  - OSEK OS : Operating System (real-time execution of ECU software and base for the other OSEK/VDX modules)

# OSEK/VDX OS Overview

- We mainly focus on OSEK OS 2.2.2 in this section.
  - A single processor operating system.
  - A static RTOS where all kernel objects are created at compile time.
- Mechanisms :
  - Concurrent execution mechanisms
    - Task
      - » A task provides the framework for the execution of functions
    - Interrupt
      - » Mechanism for processing asynchronous events
    - Alarm & Counter
      - » Mechanisms for processing recurring events
  - Synchronization mechanisms
    - Event
      - » Mechanism for concurrent processing synchronization
    - Resources
      - » Mechanism for mutual concurrent access exclusion

# OSEK/VDX Task overview

## ● Semantics:

- A task provides the framework for the execution of functions. The scheduler organizes the sequence of task execution.
- Specific Properties :
  - **Priority**: UINT32
  - **StackSize**: UINT32
- Specific Services :
  - **ActivateTask (TaskID)** : The task <TaskID> is transferred from the *suspended* state into the *ready* state
  - **ChainTask (TaskID)** : This service causes the termination of the calling task. After termination of the calling task a succeeding task <TaskID> is activated.

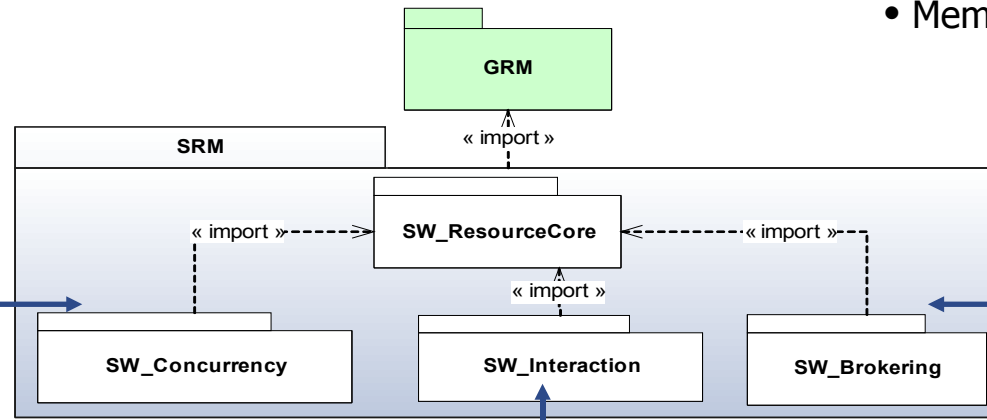
# What is supported by the SRM profile ?

## Concurrent execution contexts:

- Schedulable Resource (Task)
- Memory Partition (Process)
- Interrupt Resource
- Alarm

## Hardware and software resources brokering:

- Drivers
- Memory management

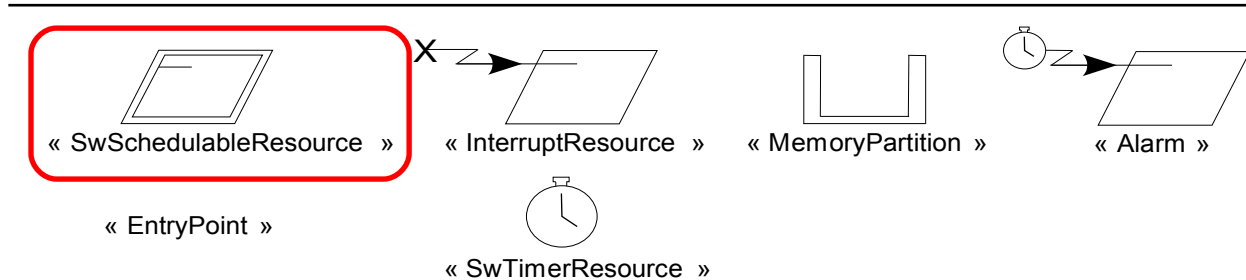


## Interactions between concurrent contexts:

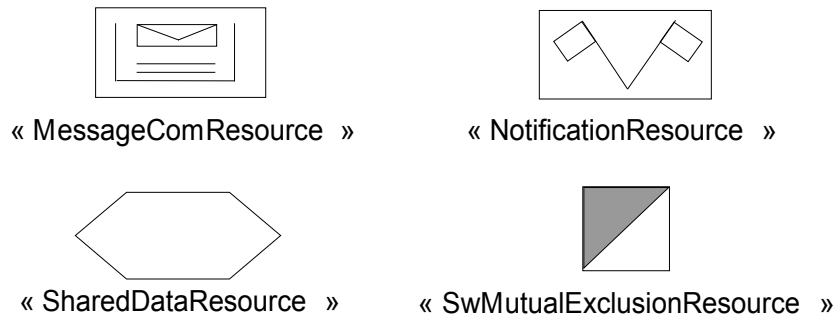
- Communication (Data exchange)
  - ✓ Shared data
  - ✓ Message (Message queue)
- Synchronization
  - ✓ Mutual Exclusion (Semaphore)
  - ✓ Notification (Event mechanism)

# Overview of the UML extensions for SRM

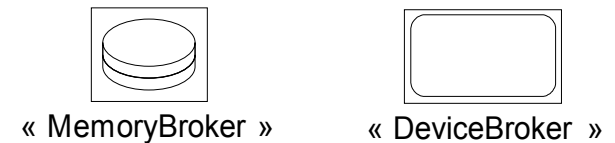
## SRM::SW\_Concurrency



## SRM::SW\_Interaction

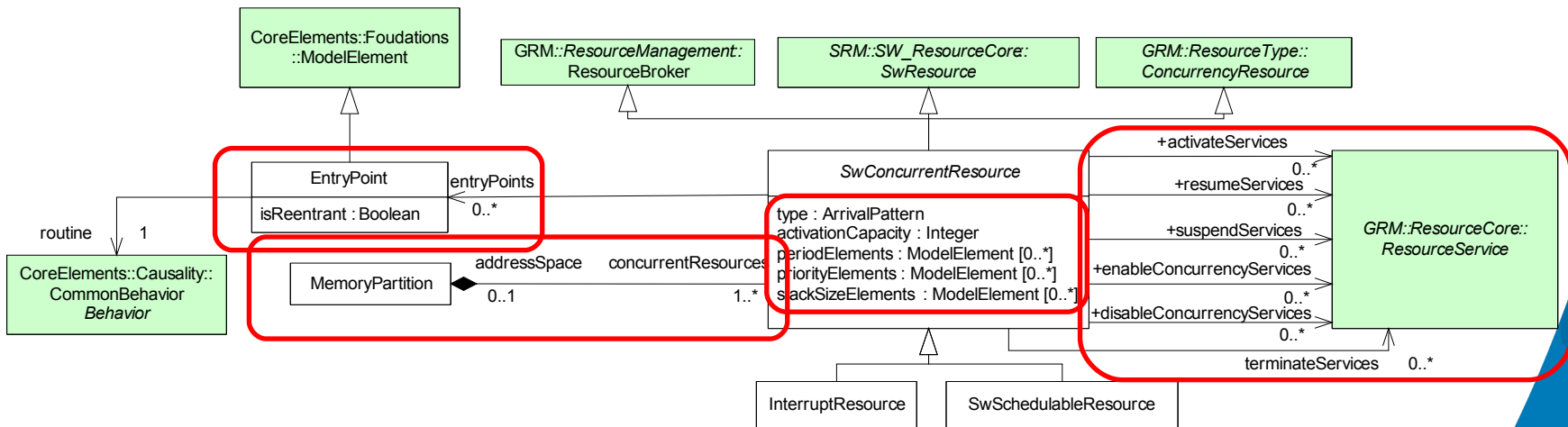


## SRM::SW\_Brokering



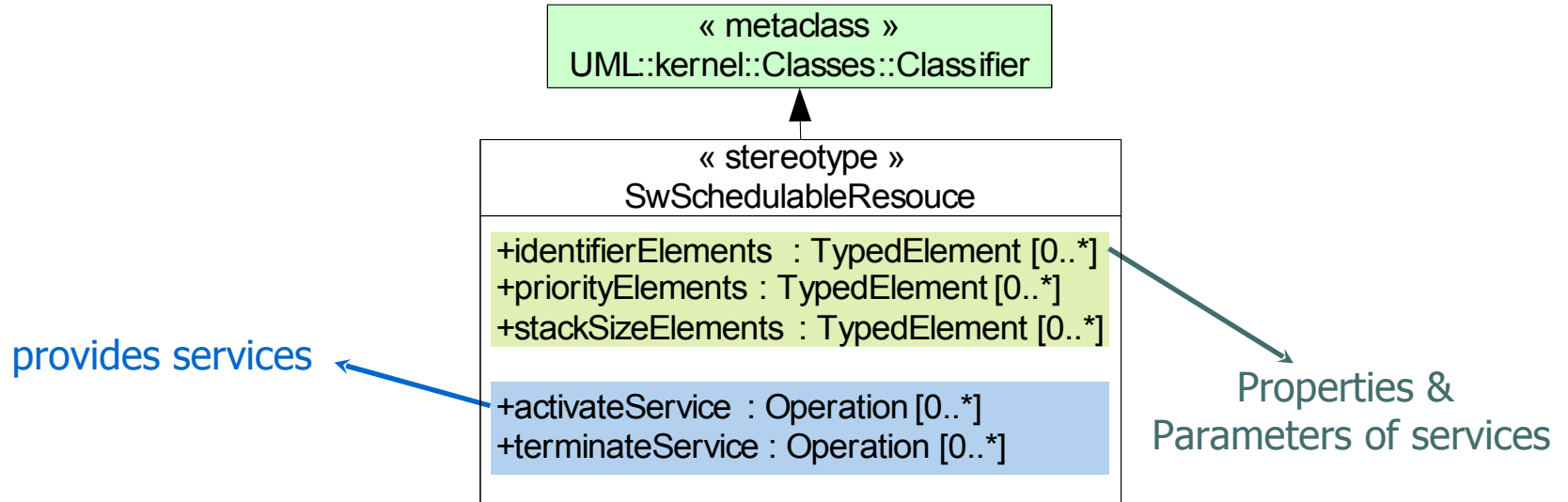
# Details of the SRM::SwSchedulableResource stereotype

- SwSchedulableResource (from MARTE::SR::Concurrency package)
  - Semantic :
    - Resources which execute concurrently to other concurrent resource
    - Periodic or aperiodic
  - Main features
    - Owns an entry point
      - Code to execute in its execution context
    - May be restrict to a specific address space (i.e. a memory partition)
    - Owns properties : Priority, Deadline, Period, StackSize ...
    - Provides services : Activate, Resume, Suspend ...



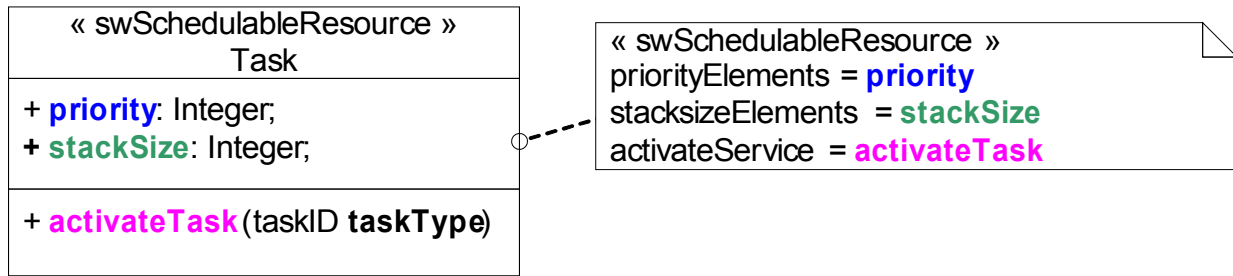


# How to model the OSEK/VDX Task with SwSchedulableResource ?



## MARTE profile view

## User model view

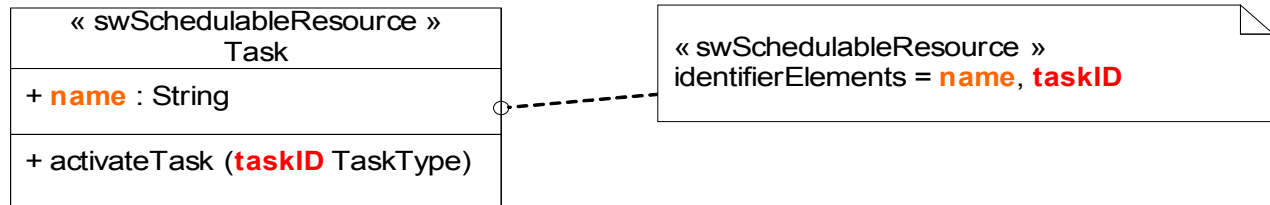


# SRM modeling possibilities

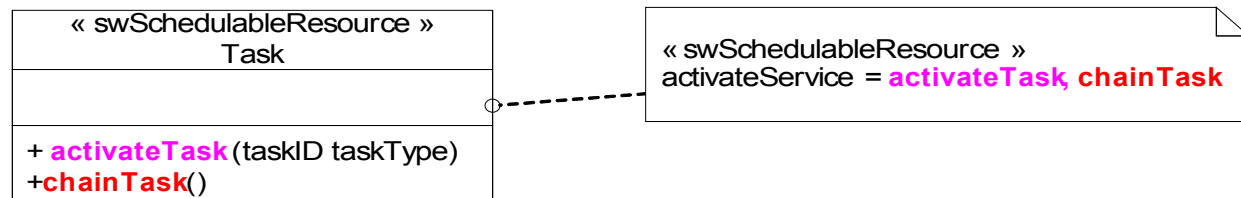
- How to model multiples candidates for the same semantic ?
  - Answer : All stereotype tags have multiple multiplicities. Thus, it is possible to reference multiple candidates for the same tag.

- Examples

- The name and the *taskID* own the same semantic : the task identifier



- Both *activateTask* and *chainTask* services activate a task

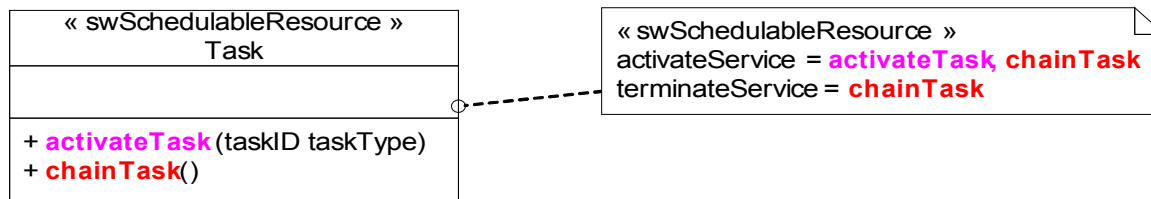


# SRM modeling possibilities

- How to model a feature which have multiple semantic ?
  - Answer : Feature can be referenced by several tags

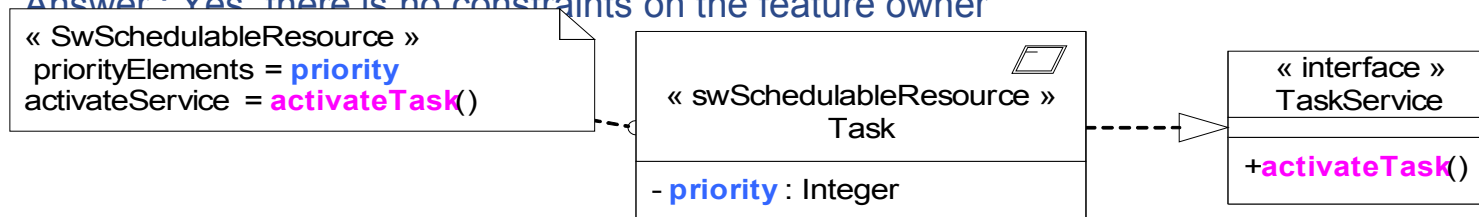
- Example

- The *chainTask* service terminate the calling task and activate the <taskID> one



- Is it possible to reference a feature even if the feature owner is not the stereotyped element ?

- Answer : Yes, there is no constraints on the feature owner



- SRM allows multiple usages

- User can use constraints, such as OCL rules, to limit those possibilities

# OSEK/VDX OS Overview

- We mainly focus on OSEK OS 2.2.2 in this section.
  - A single processor operating system.
  - A static RTOS where all kernel objects are created at compile time.
- Mechanisms :
  - Concurrent execution mechanisms
    - Task
      - » A task provides the framework for the execution of functions
    - Interrupt
      - » Mechanism for processing asynchronous events
    - Alarm & Counter
      - » Mechanisms for processing recurring events
  - Synchronization mechanisms
    - Event
      - » Mechanism for concurrent processing synchronization
    - Resources
      - » Mechanim for mutual concurrent access exclusion

# OSEK/VDX Event

## ● Semantics:

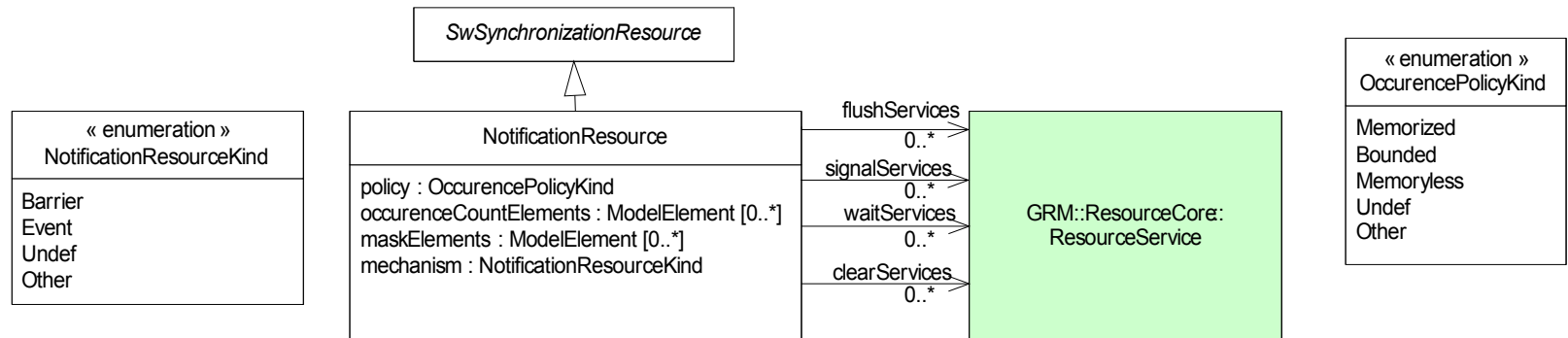
- The event mechanism
  - is a means of synchronisation
  - initiates state transitions of tasks to and from the *waiting* state.
- Specific Properties :
  - **Mask** : EventMaskType
- Specific Services :
  - **setEvent (TaskID, Mask)** : The events of task **<TaskID>** are set according to the event mask **<Mask>**. Calling **SetEvent** causes the task **<TaskID>** to be transferred to the ready state, if it was waiting for at least one of the events specified in **<Mask>**.
  - **waitEvent (Mask)** : The state of the calling task is set to *waiting*, unless at least one of the events specified in **<Mask>** has already been set.
  - ...

# OSEK/VDX Event as a NotificationResource

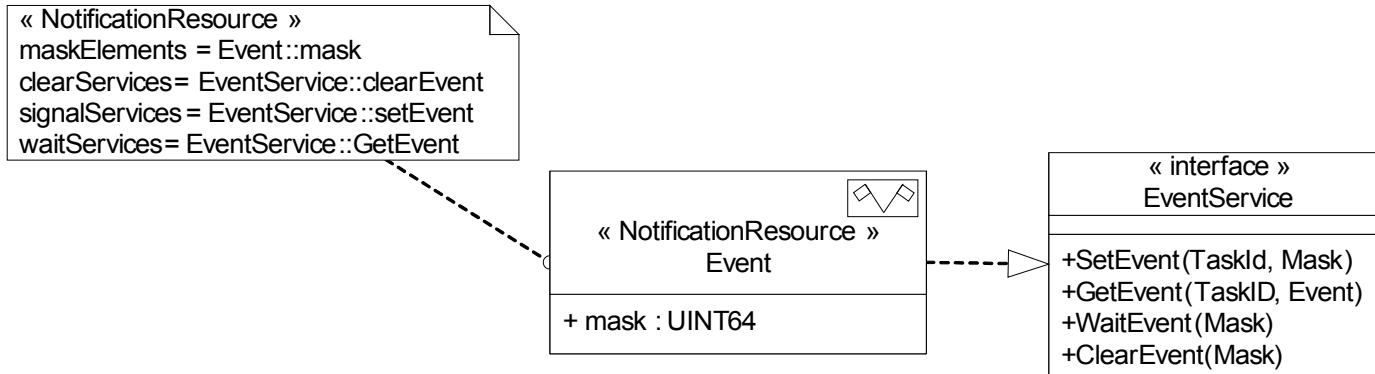
## NotificationResource

– Semantic :

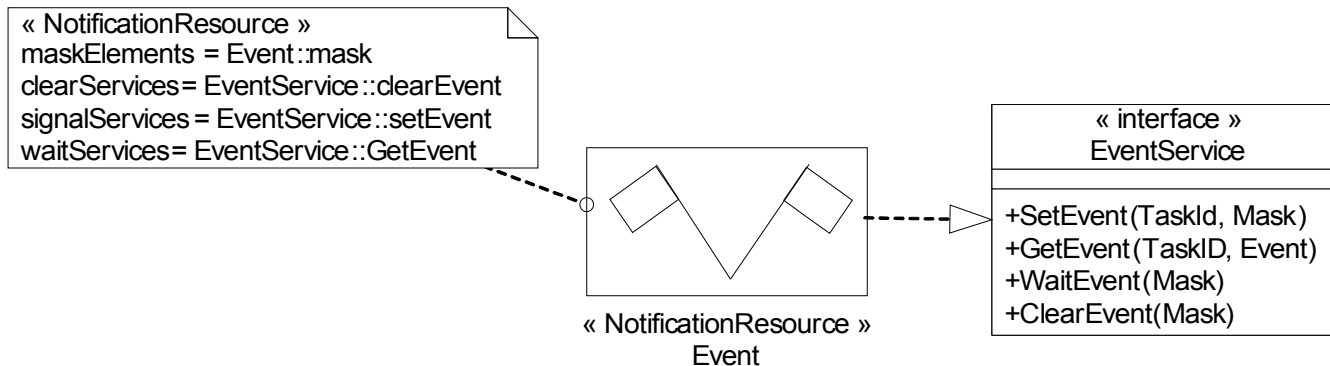
- *NotificationResource* supports control flow by notifying the occurrences of conditions to awaiting concurrent resources Linked to an entry point (code to execute in its execution context)



# OSEK/VDX Event as a NotificationResource



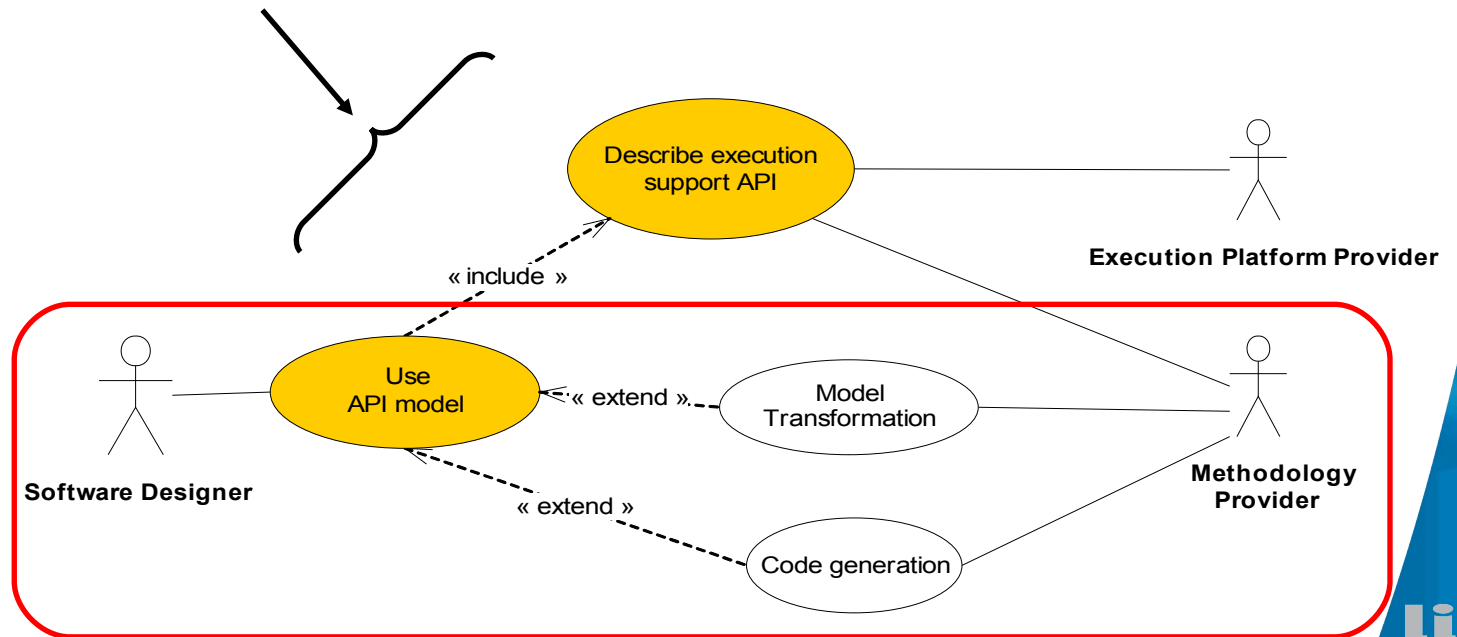
(i) Stereotype icon



(ii) Stereotype shape

# In which typical cases shall I use SRM ?

**Software Resource Modeling (SRM)**

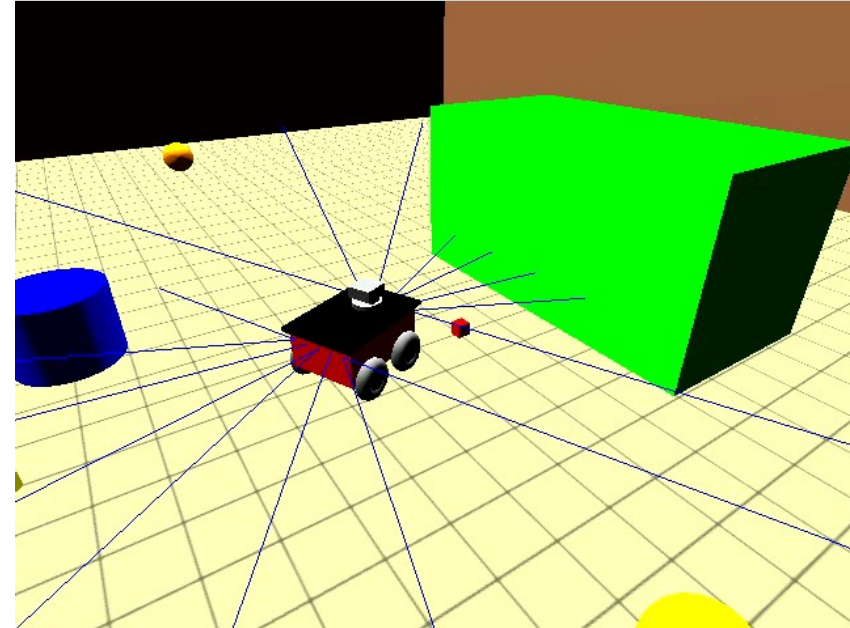


## Examples of RTOS API model uses

- Example 1: Model-based design of multitask applications
  - Illustrated on a robot controller application
- Example 2: OS configuration file generation
  - Generation of the OSEK OIL configuration files
- Example 3: Assistance to port applications
  - From OSEK to ARINC multitask design

# Case study: A simple robot controller software

- Goal :
  - A motion controller system for an exploration autonomous mobile robot.
- Robot features :
  - Pioneer Robot (P3AT)
    - Four driving wheels
    - A camera
    - Eight sonar sensors ...
- Controller features
  - Motion Controller
    - Two periodic tasks :
      - Acquisition : Get sonar sensors interfaces
        - » period = 1 ms
      - trajectoryController : Set new speed
        - » period = 4 ms
    - OSEK/VDX execution support
      - Trampoline (<http://trampoline.rts-software.org/>)



Robot Simulator  
<http://playerstage.sourceforge.net/gazebo/gazebo.html>

list

cei

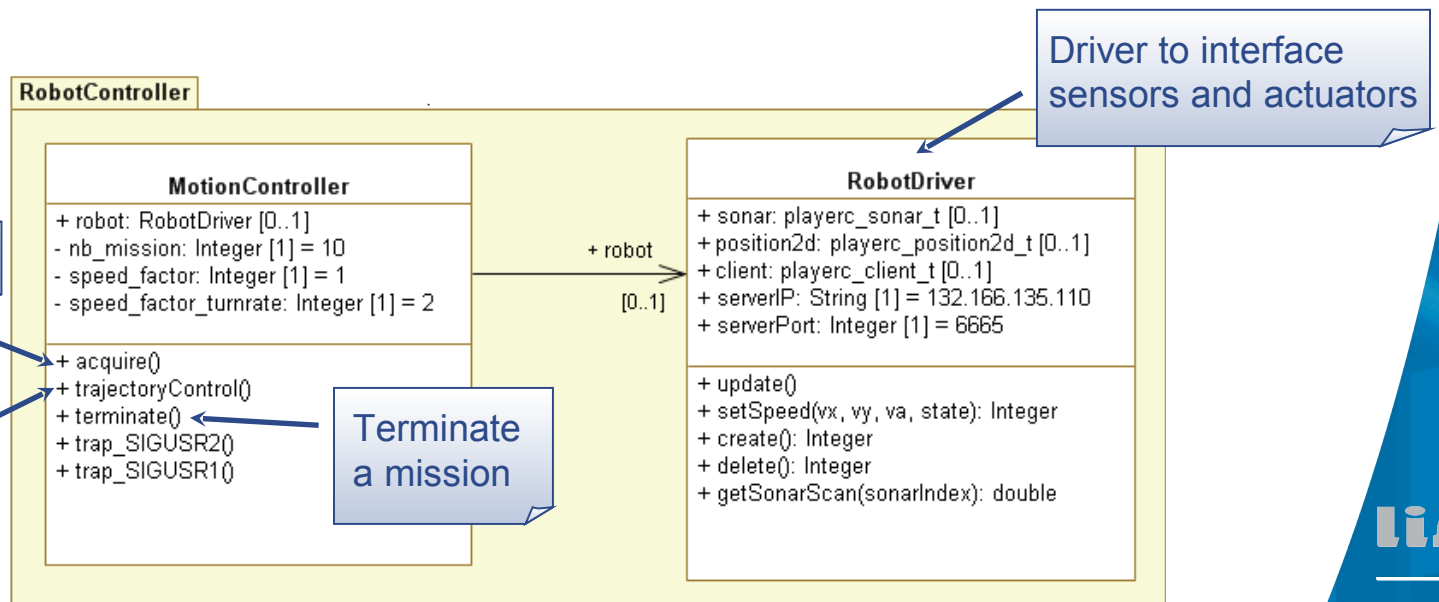


# Purpose

- Provide a multitask design of our robot controller
  - Multitask design must be based on the OSEK/VDX platform
- Design process
  - Platform Provider supplies the OSEK/VDX model library
    - model library is described with the SRM Profile (previous slides)
  - User designs a multitask model of the application
    - High level application description
      - Deduce tasks requirements
    - Multitask design
      - User instantiates model library classifiers to design the multitask application
    - Binding of the high level description with the multitask design

# Application design

- A robot controller entity :
  - To control the robot motions
  - A multitask entity
    - Acquire the sonar
    - Compute and assign new speed order
- A robot driver : To interface robot sensors and actuators



# Basic multitask model

## ● Motion Controller

– Two periodic tasks :

- Acquisition :

- Entry point : `acquire()` (Get sonar sensors interfaces)

- Periodic

- period = 1 ms

- trajectoryController

- Entry point : `trajectoryControl()` (Compute and assign new speed order)

- Periodic

- period = 4 ms

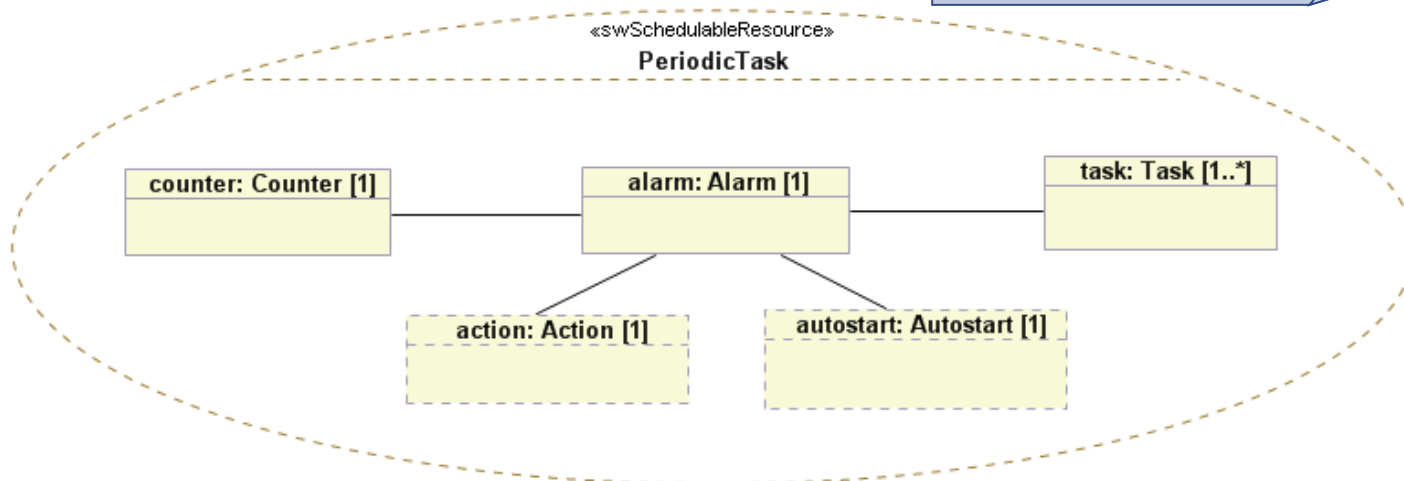
## ● Robot driver

– No tasks

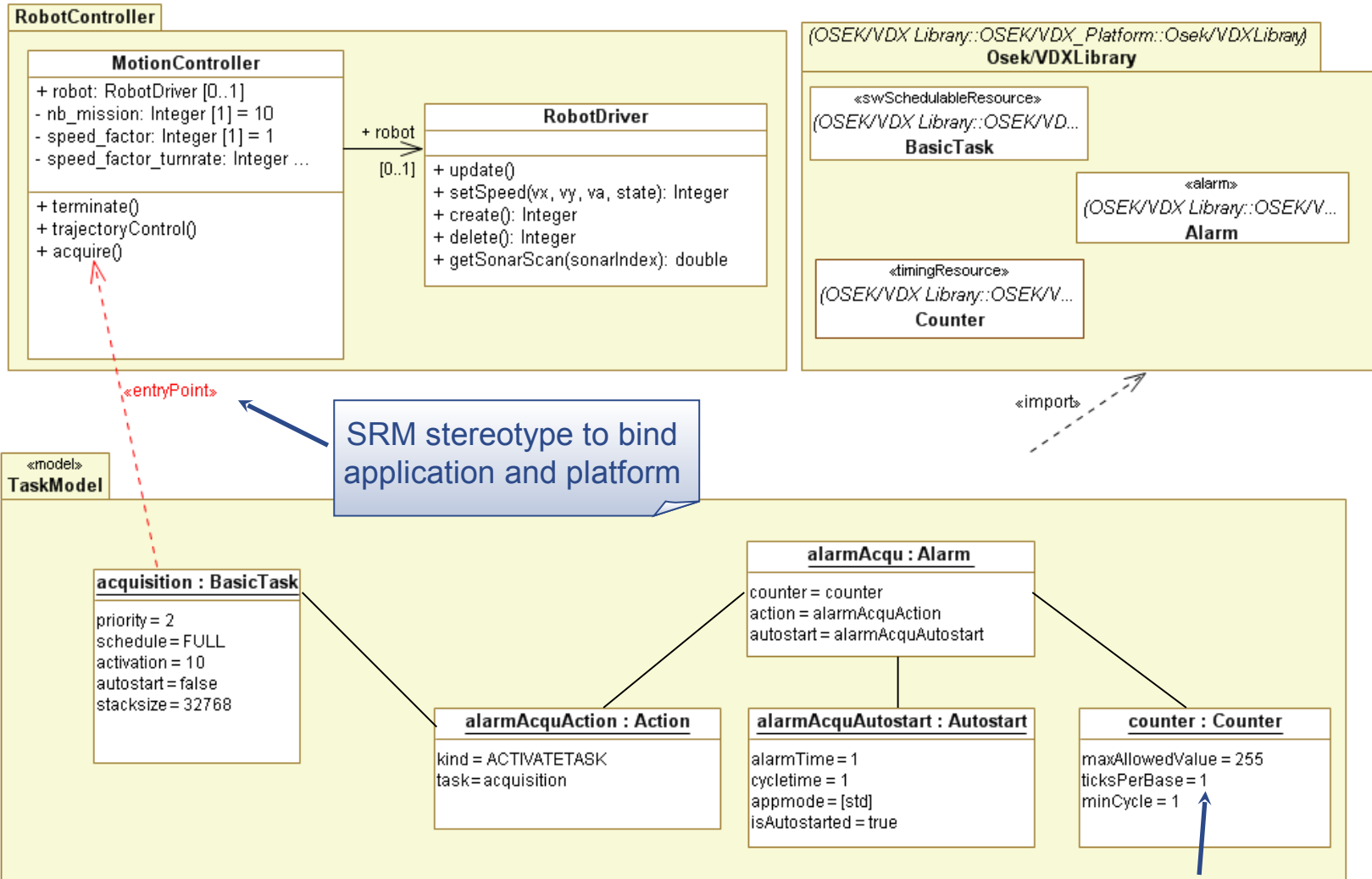
# Periodic task in OSEK/VDX

- OSEK/VDX periodic task pattern
  - One OSEK/VDX Counter
    - Period : periodic task Period
  - One OSEK/VDX Task
    - Entry Point : periodic task Entry Point
  - One OSEK/VDX Alarm
    - AutoStart : Triggered by the counter
    - Action : Activate the task

SRM Profile is used to describe the pattern



# Basic Robot Controller task models



Period of the periodic task acquisition : 1 ms

# Use case : OSEK Configuration File generation

## Purpose

- Generate of the OSEK OIL configuration files from the multi-task design of the robot controller

## OIL: OSEK Implementation Language

- <http://osek-vdx.org>
- The goal of OIL is to provide a mechanism to configure an OSEK application inside a particular CPU
- Principle
  - For each CPU, there must be an OIL description
  - All OSEK system objects are described using OIL objects
  - OIL descriptions may be :
    - hand-written
    - or generated by a system configuration tool

```
OIL_VERSION = "2.5" : "RobotController" ;

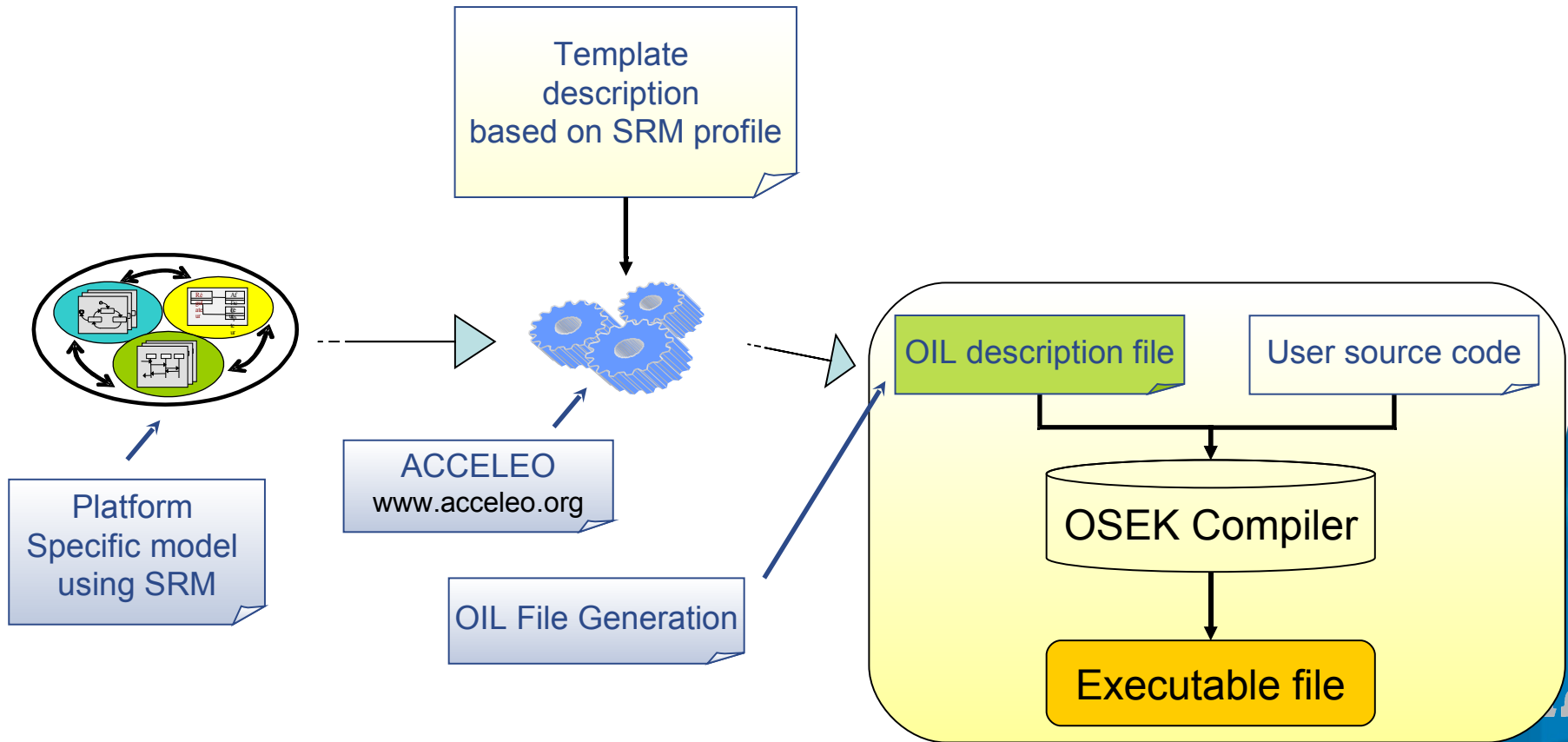
IMPLEMENTATION OSEK {
};

CPU cpu {
  APPMODE std {
  };

  COUNTER counter {
    MAXALLOWEDVALUE = 255 ;
    TICKSPERBASE = 1 ;
    MINCYCLE = 1 ;
  };
  ALARM alarmAcqu {
    COUNTER = counter ;
    ACTION = ACTIVATETASK {
      TASK = acquisition ;
    };
    AUTOSTART = TRUE {
      ALARMTIME = 1 ;
      CYCLETIME = 1 ;
      APPMODE = std ;
    };
  };
};

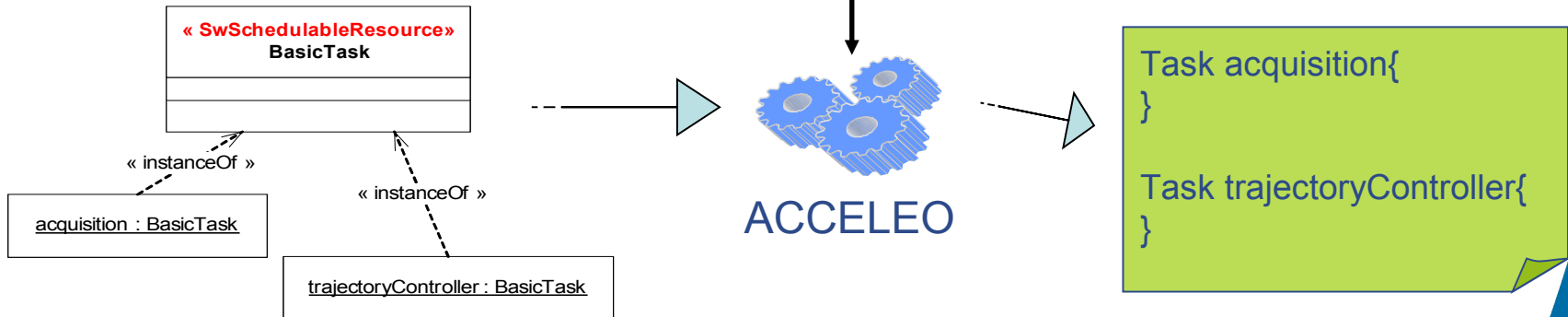
TASK acquisition {
  PRIORITY = 2 ;
  SCHEDULE = FULL ;
  ACTIVATION = 10 ;
  AUTOSTART = FALSE ;
  STACKSIZE = 32768 ;
};
...
```

# Principle to generate from a UML model an OIL description file



# OIL template examples

```
For each UML InstanceSpecification {  
  if its classifier has the SRM SwSchedulableResource stereotype then {  
    generateText {  
      Task <Instance Name>{}  
    }endGenerate  
  }endif  
}
```

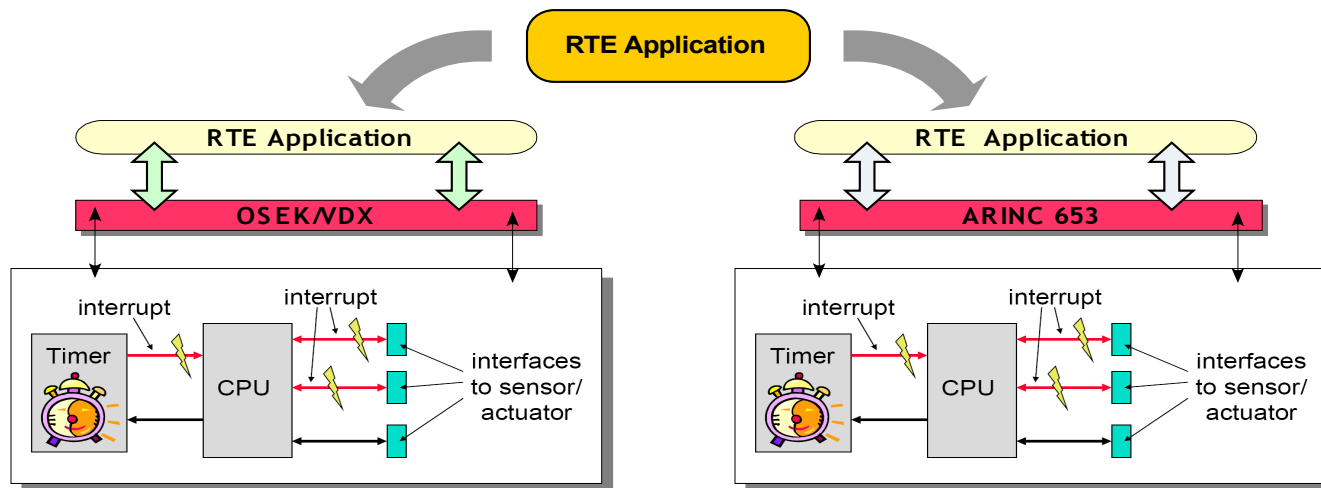


# Generation of the OIL file in the Papyrus UML Tool

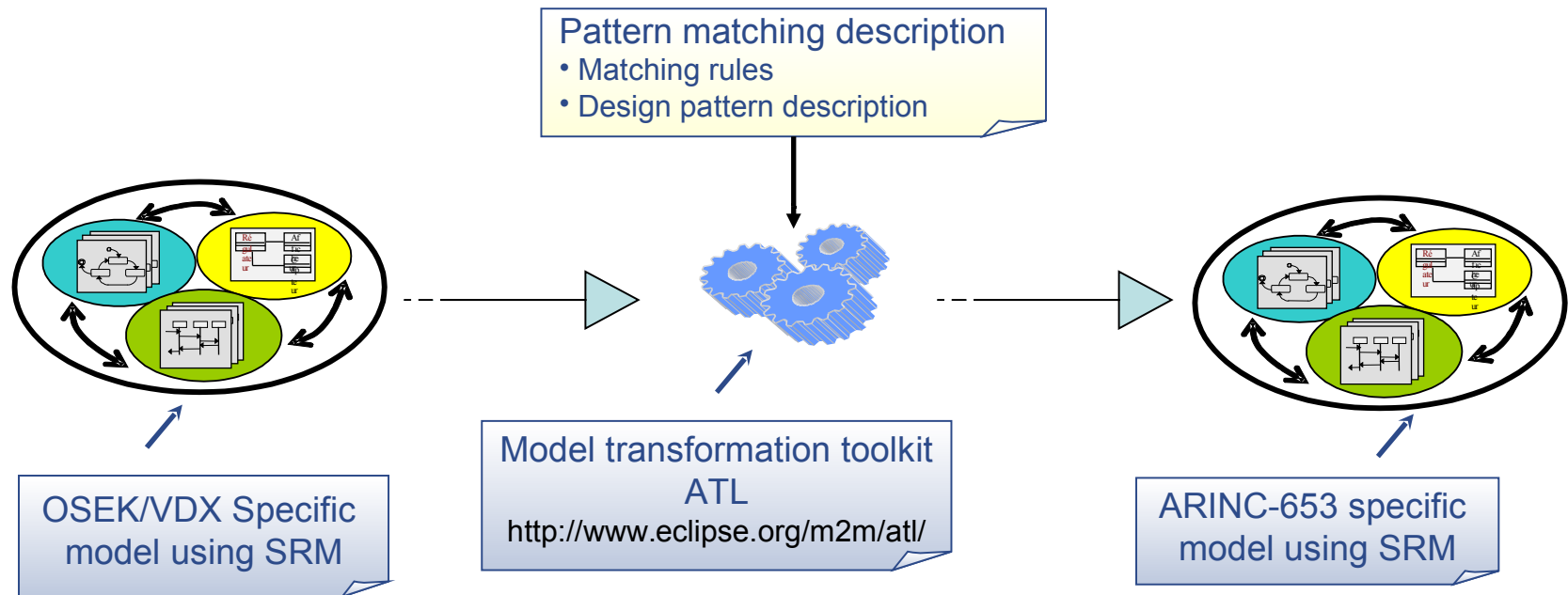
## Example 3 : Assist user to port multitask designs

### ● Purpose:

- Assist user to port the multitask design to an ARINC-653 RTOS
  - ARINC 653 standard provides avionics application software with the set of basic services to access the operating system and other system-specific resources.

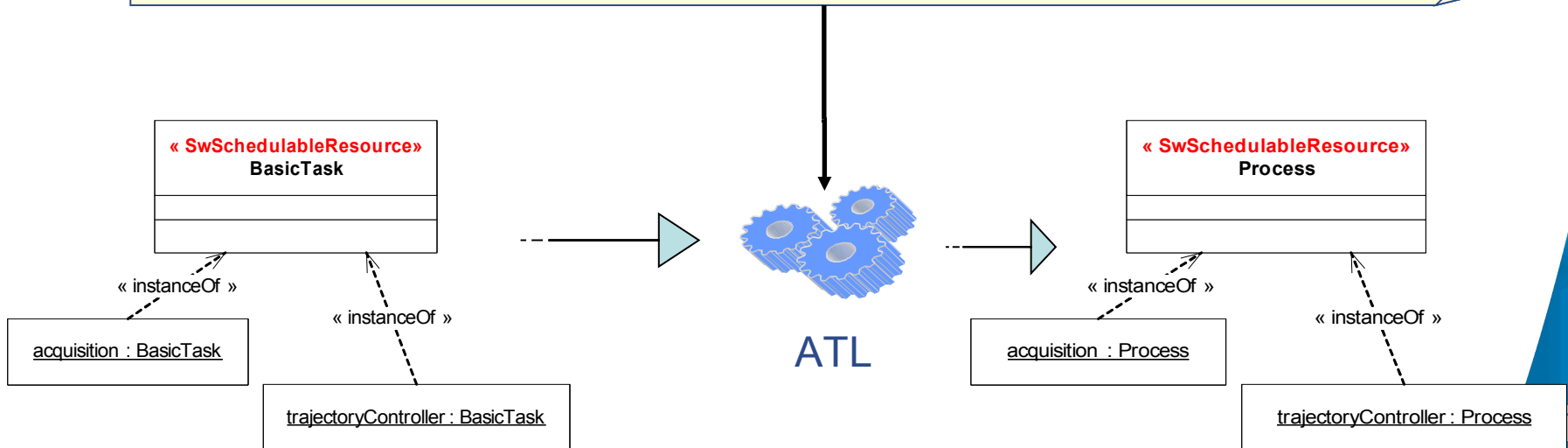


# Assist user to port multitask designs



# Matching pattern example (1/2)

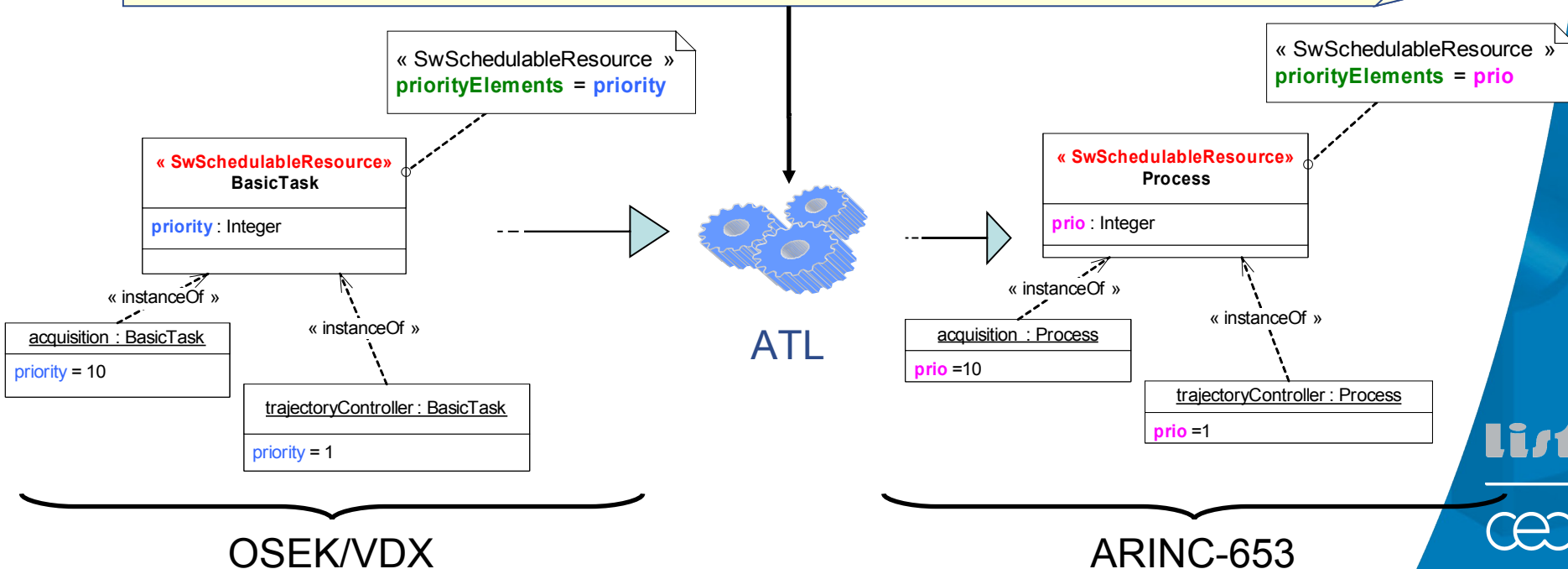
```
For each UML InstanceSpecification {  
    if its classifier has the SRM SwSchedulableResource stereotype then {  
        • generate a new Instance Specification;  
        • its target classifier is that which is stereotyped  
          SwSchedulableResource in the target execution support;  
    }endGenerate  
}endif  
}
```



# Matching pattern example (2/2)

```

For each UML InstanceSpecification {
  if its classifier has the SRM SwSchedulableResource stereotype then {
    • ...
    • each source priorityElements match one target priorityElements
  }endGenerate
}endif
}
    
```



# Assist user to port multi-task designs in the Papyrus UML tool : a basic example

## Related resources

- The official MARTE web site: [www.omgmarTE.org](http://www.omgmarTE.org)
  - Tutorials, events, projects related and tools
- [www.papyrusuml.org](http://www.papyrusuml.org)
  - On open source Eclipse plug-in for UML2 graphical modeling
  - MARTE implementation available within the V1.9 release of the tool