Accord-UML: a methodological approach for model-based development and validation of RT/E systems

Artist2 workshop: **MoCC - Models of Computation and Communication**
November 16-17, 2006

Sébastien Gérard, Arnaud Cuccuru and Frédéric Loiret

{firstName.lastName}@cea.fr
CEA-List / DTSI / SOL / L-LSP

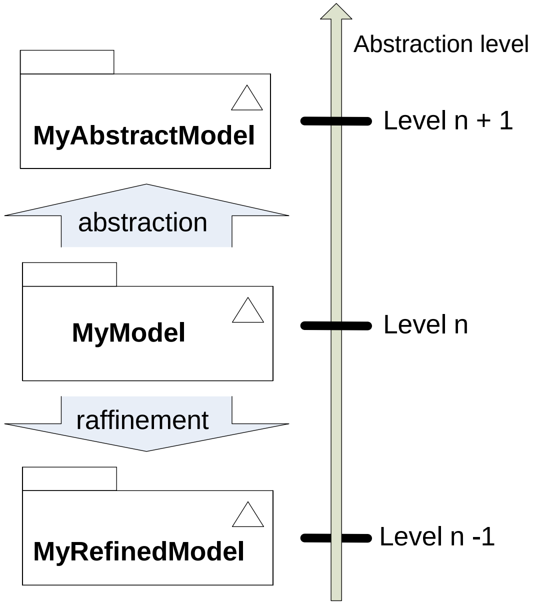
Agenda

- Context and work outlines
- The UML MoCC
- The Accord|UML proposal
- Ongoing work and next steps

MDD in a nutshell

- **Models**
  - ✓ Many definitions: *e.g.*: "An abstract (or actual) representation of an object or system from a particular viewpoint."*
  - ✓ Written with suitable modeling languages (mainly graphical)
    - → *E.g.: Ecore, MOF, EAST-ADL…*

      … UML2 and its profiles!
  - ✓ Defined through meta-models or profiles



- **Two kinds of model relationships**
  - ✓ Abstraction
    - → Need for suitable RT/E related concepts!
  - ✓ Refinement
    - → Need for specific model transformations
- **A lot of available model techniques & tools**
  - ✓ Design patterns, Aspect Oriented Modeling, Meta-modeling, Merge, Model transformations, Profiling…

* extracted from www.wikipedia.org

Abstractions issues w.r.t. RT/E-MDD

- **Well-suited concepts for modeling RT/E features**
  - ✓ RT/E quantitative features
    - → *E.g. Deadlines, WCET, Periodicity and Power consumption*
  - ✓ RT/E qualitative features
    - → Related to computation (execution)
      - » *E.g. Concurrency and synchronization*
    - → Related to communication
      - » *E.g. Synchronization modes*
- **Well-defined ("formalized") concepts**
  - ✓ RT/E models needs to be non-ambiguous models!

Needs for specific modeling languages including RT/E related artifacts with dedicated and well-defined MoCC

Refinement issues w.r.t. RT/E-MDD

- **One of the main challenge of MDD**
  - ✓ From contemplative to active role of models!

- **For refinement, active models mean mainly:**
  - ✓ Specific execution platforms for supporting RT&E-MoCC
    - → Either software or hardware (or both)
    - → *E.g. RTOS platforms such as Posix and OSEK.*
  - ✓ Dedicated model transformations to target such platforms
    - → *E.g. RT/E design patterns and code generation*

UML is the de facto standard for MDD:
Accord|$_{\text{UML}}$, a UML-based approach
for RT&ES development

Agenda

- Context and work outlines
- **The UML MoCC**
- The Accord|UML proposal
- Ongoing work and next steps

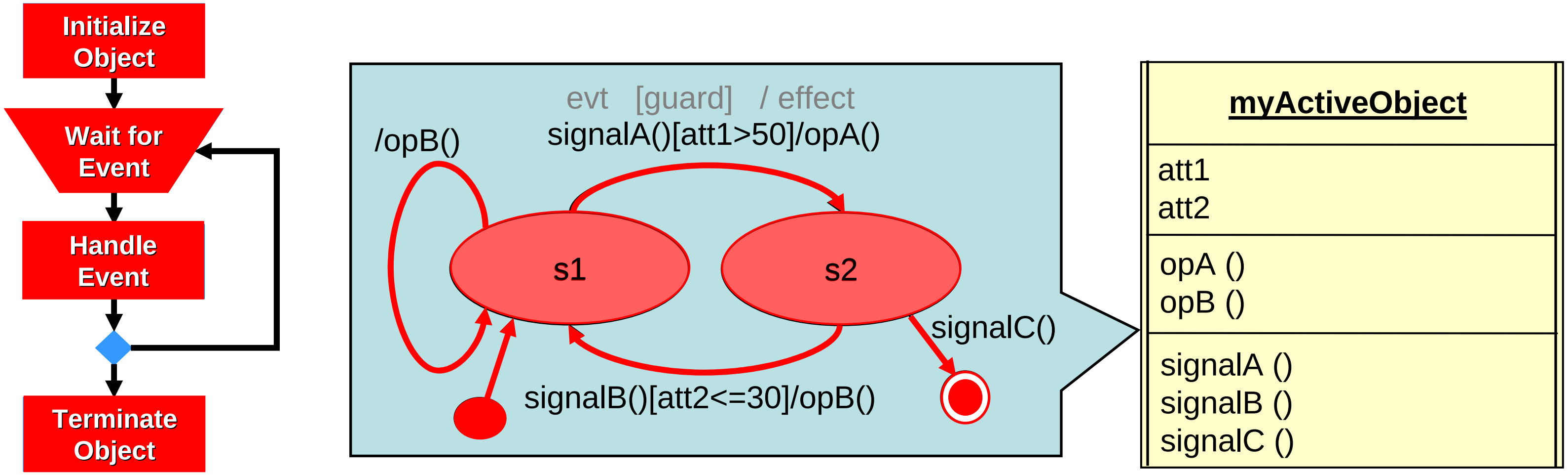
Outlines of MoCC within UML

- **A UML model = { objects with behavior and communicating by message passing }**
- **UML models of communication**
  - ✓ Operation-based message
    - → Synchronous or asynchronous / With input, output or returns parameters / Point-to-point
  - ✓ Signal-based message
    - → Asynchronous / With input parameters / Broadcast or multicast
- **UML models of computation**
  - ✓ Active objects (concurrent unit of UML)
    - → Have their own thread of control
    - → Their behavior determine the response to communications
  - ✓ Passive objects
    - → Computation resource of a caller active object to execute
    - → Concurrency policies on provided services
      - » Sequential, guarded and concurrent
- **Semantics variation points of UML**
  - ✓ Parts of the specification that are open
    - → Ex1. Signals may broadcased or mulitcasted
    - → Ex2. Statemachine have a queue that may be FIFO, LIFO, Mailbox…
  - ✓ May be considered as parameters of a generic MoCC
  - ✓ Needs to fixed within a dedicated UML profile

The UML profile for Accord|$_{\text{UML}}$ is a such a profile!

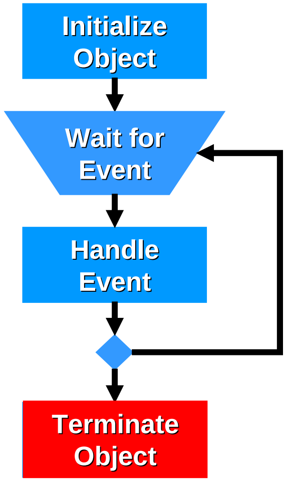
MoCC of Active Object behaviored with statemachine

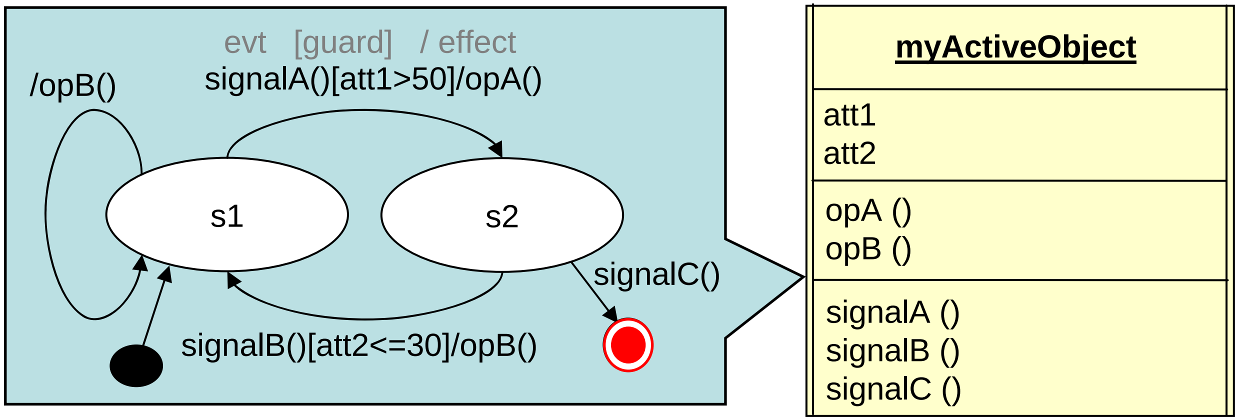
- Run-to-completion semantics of the statemachine
  - ✓ A four steps cyclic process
    - → Object initialization
    - → Object waiting for events
    - → Object handling an event
    - → Object termination



MoCC of Active Object behaviored with statemachine

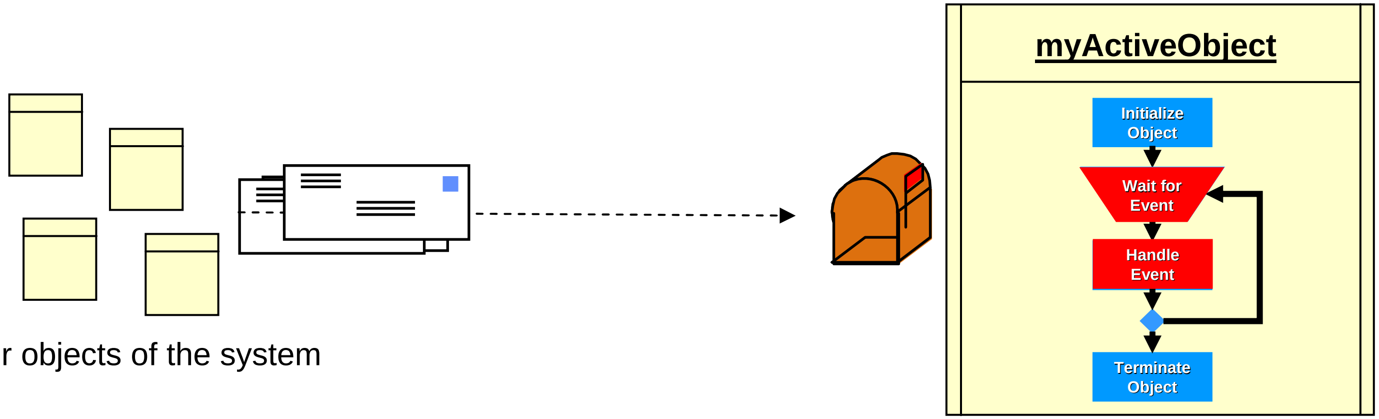
- **Run-to-completion semantics of the statemachine**
  - ✓ A four steps cyclic process
    - → Object initialization
    - → Object waiting for events
    - → Object handling an event
    - → Object termination





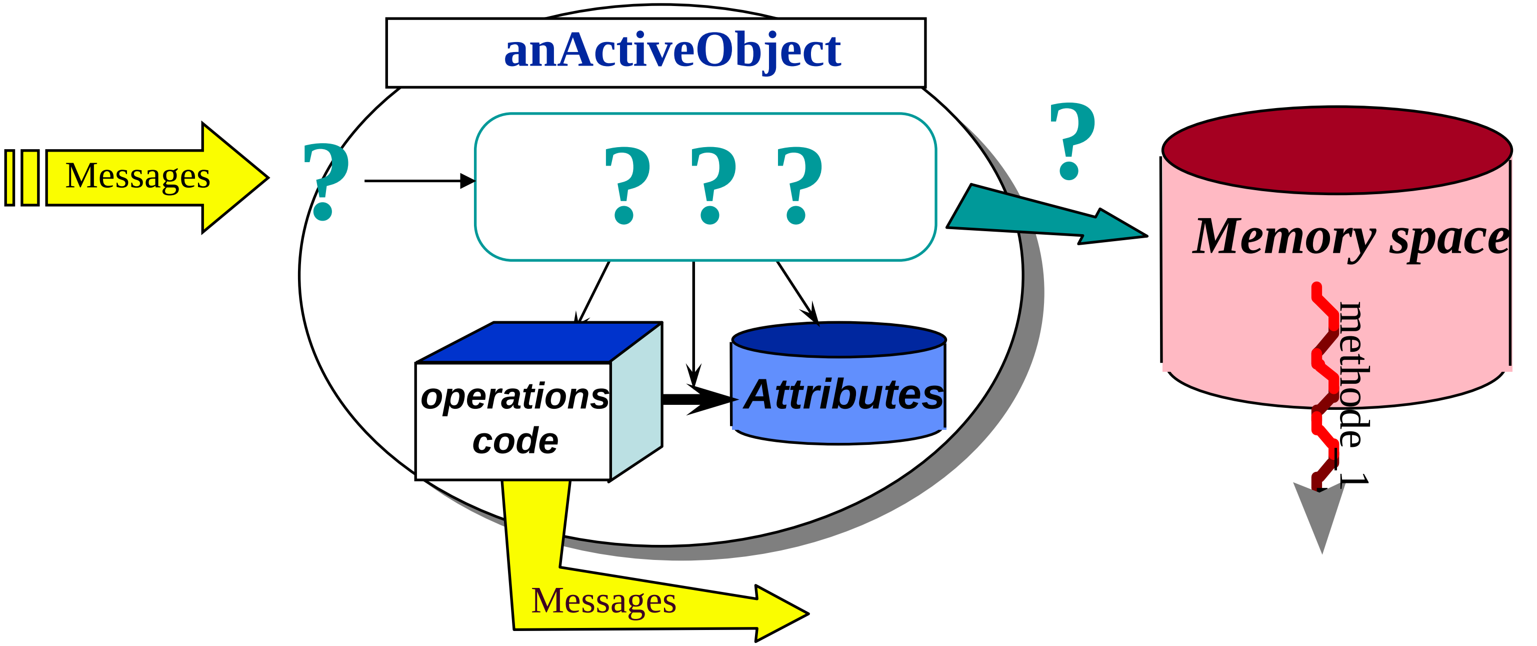
MoCC of Active Object behaviored with statemachine

- **Focus on active object dynamics:**
  - ✓ A four steps cyclic process
    - → Object initialization
    - → Object waiting for events
    - → Object handling an event
    - → Object termination
  - ✓ Basic execution sketch



Other objects of the system

Summary of UML Active Objects

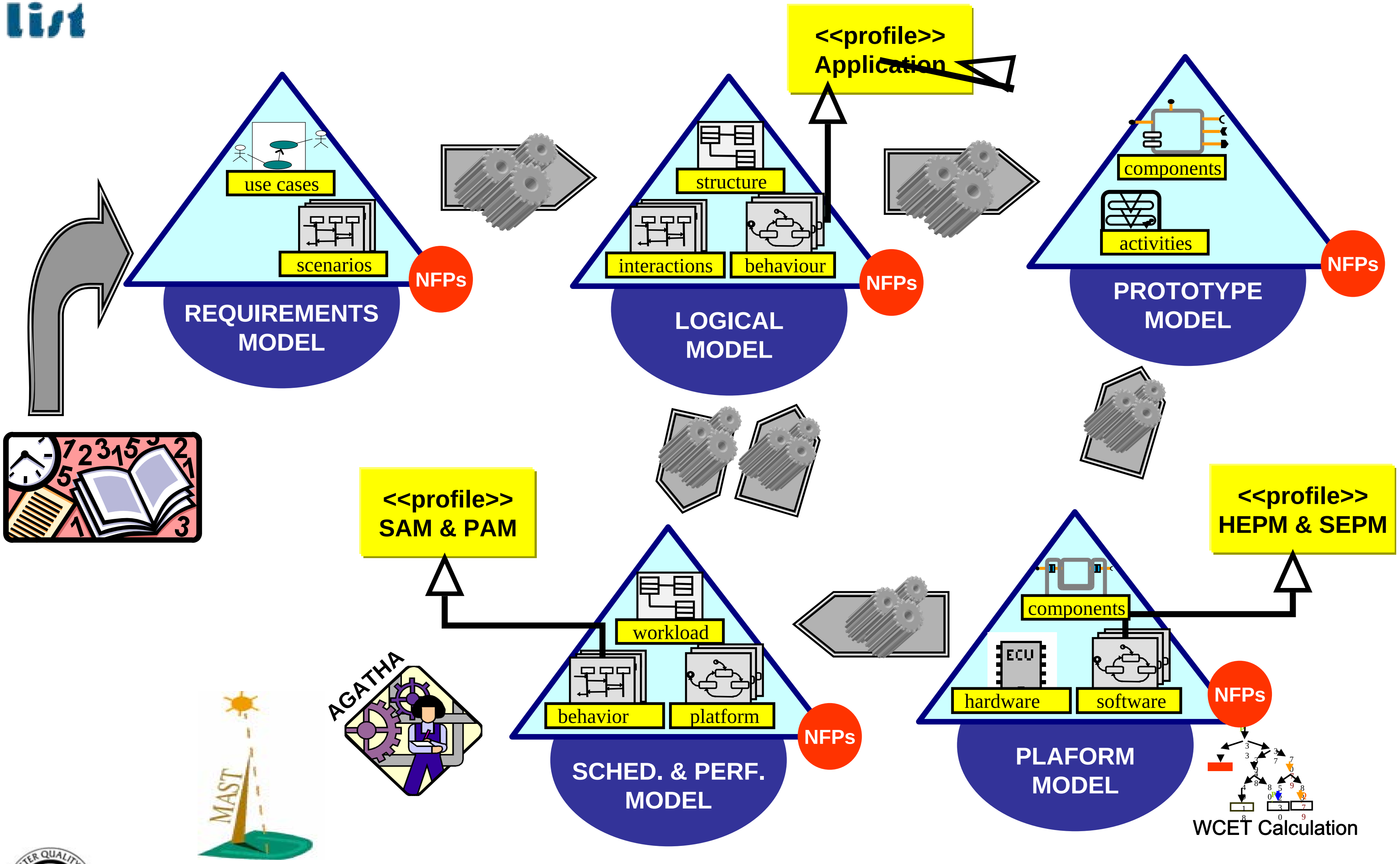


- **Main characteristics of the UML active object**
  - One single active resource ?
  - not well-defined behavior semantics

Agenda

- Context and work outlines
- The UML MoCC
- The Accord|UML proposal
- Ongoing work and next steps

Outlines of the Accord|$_{\text{UML}}$ methodological framework



<<profile>>
Application

use cases

scenarios

NFPs

REQUIREMENTS MODEL

structure

interactions

behaviour

NFPs

LOGICAL MODEL

components

activities

NFPs

PROTOTYPE MODEL

<<profile>>
SAM & PAM

workload

behavior

platform

NFPs

SCHED. & PERF. MODEL

<<profile>>
HEPM & SEPM

components

hardware

software

NFPs

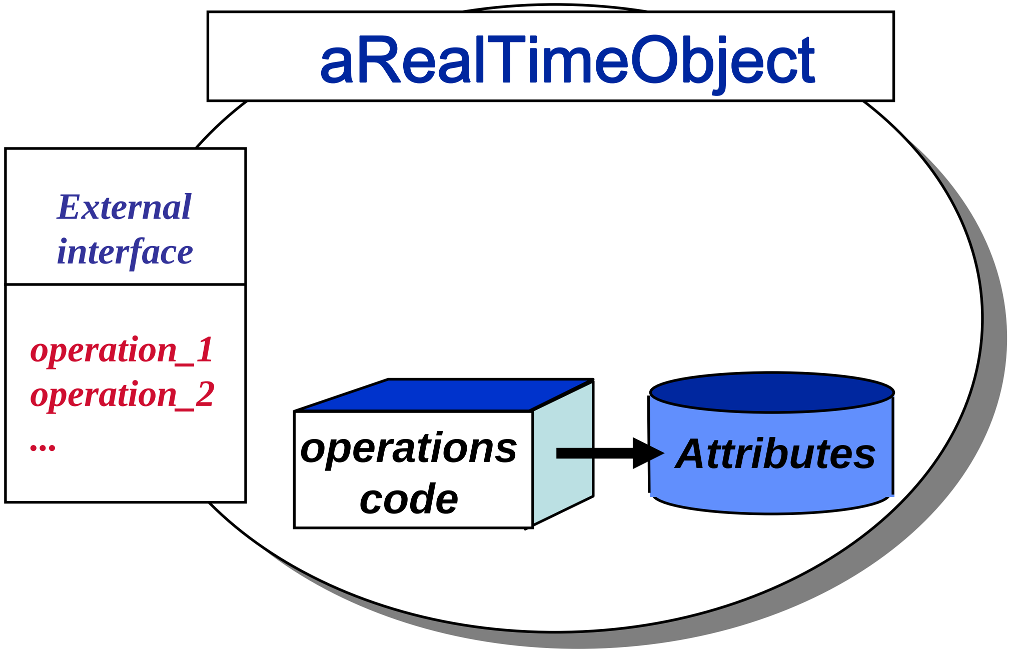
PLAFORM MODEL

WCET Calculation

The ACCORD Real-Time Objects MoCC

User point of view :

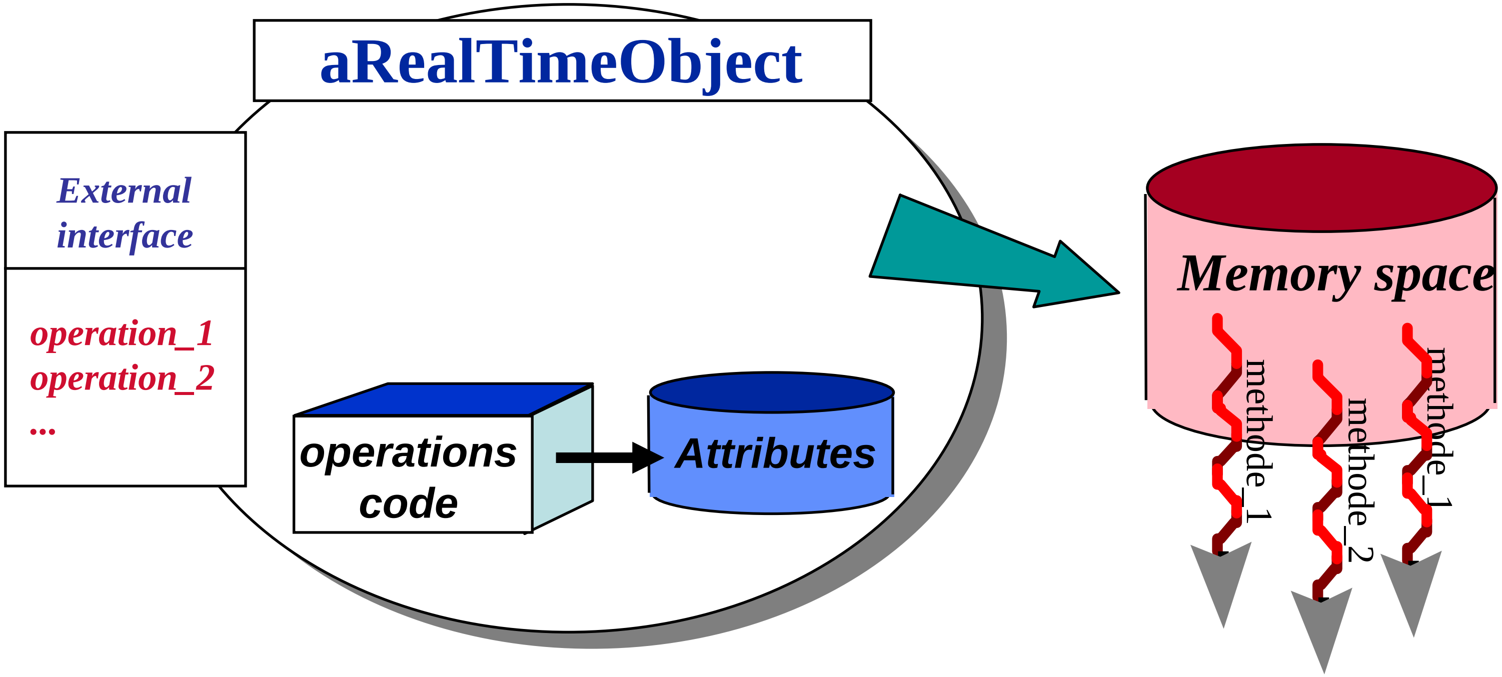
an object encapsulating data & processing



The ACCORD Real-Time Objects MoCC

User point of view :

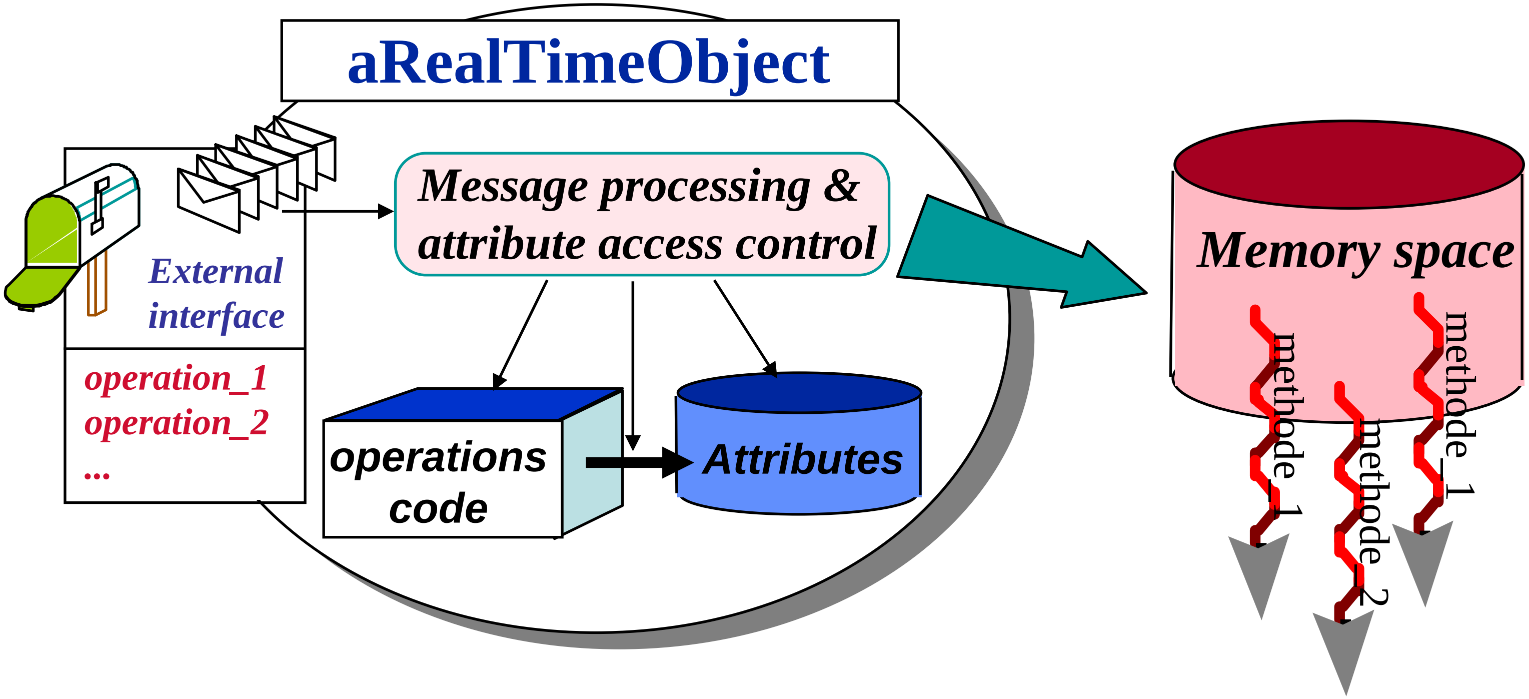
an object with its own processing resources



The ACCORD Real-Time objects MoCC

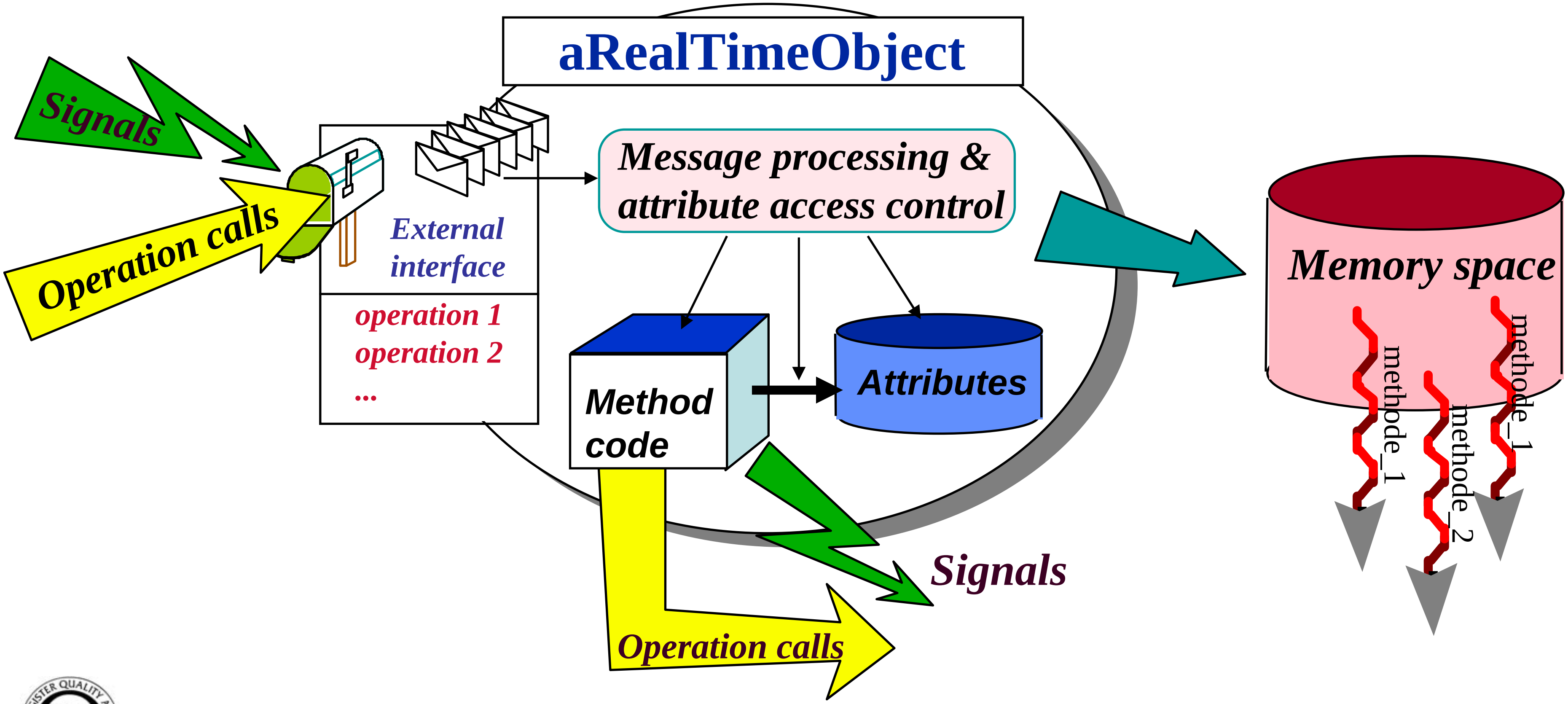
User point of view :

an object performing itself the control of its processing

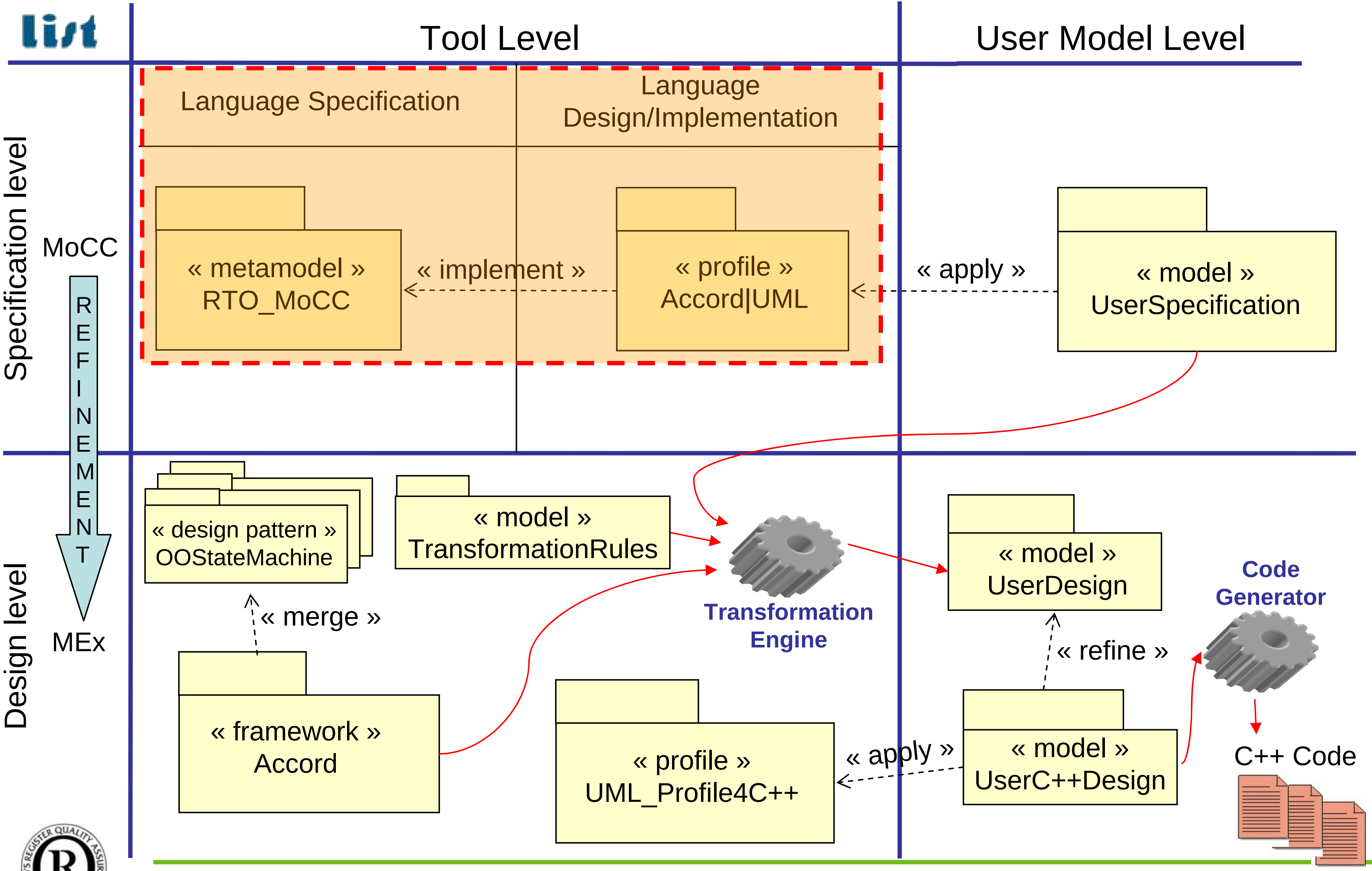


The ACCORD Real-Time objects MoCC

User point of view : *an autonomous computing entity with a standard UML object interface*

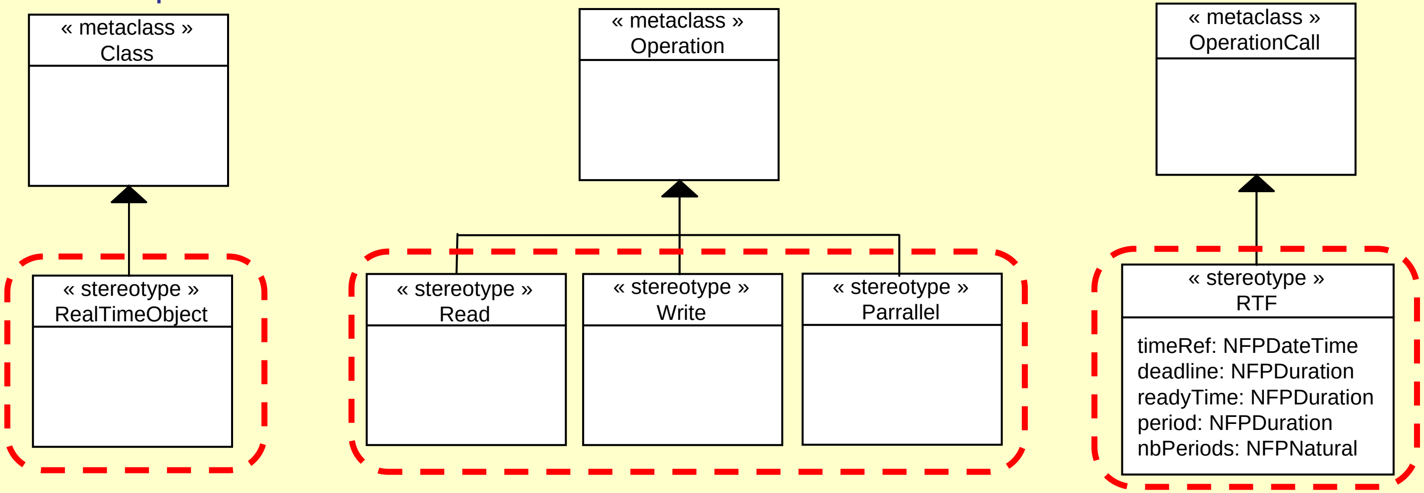


MoCC and MEx within Accord|$_{\text{UML}}$



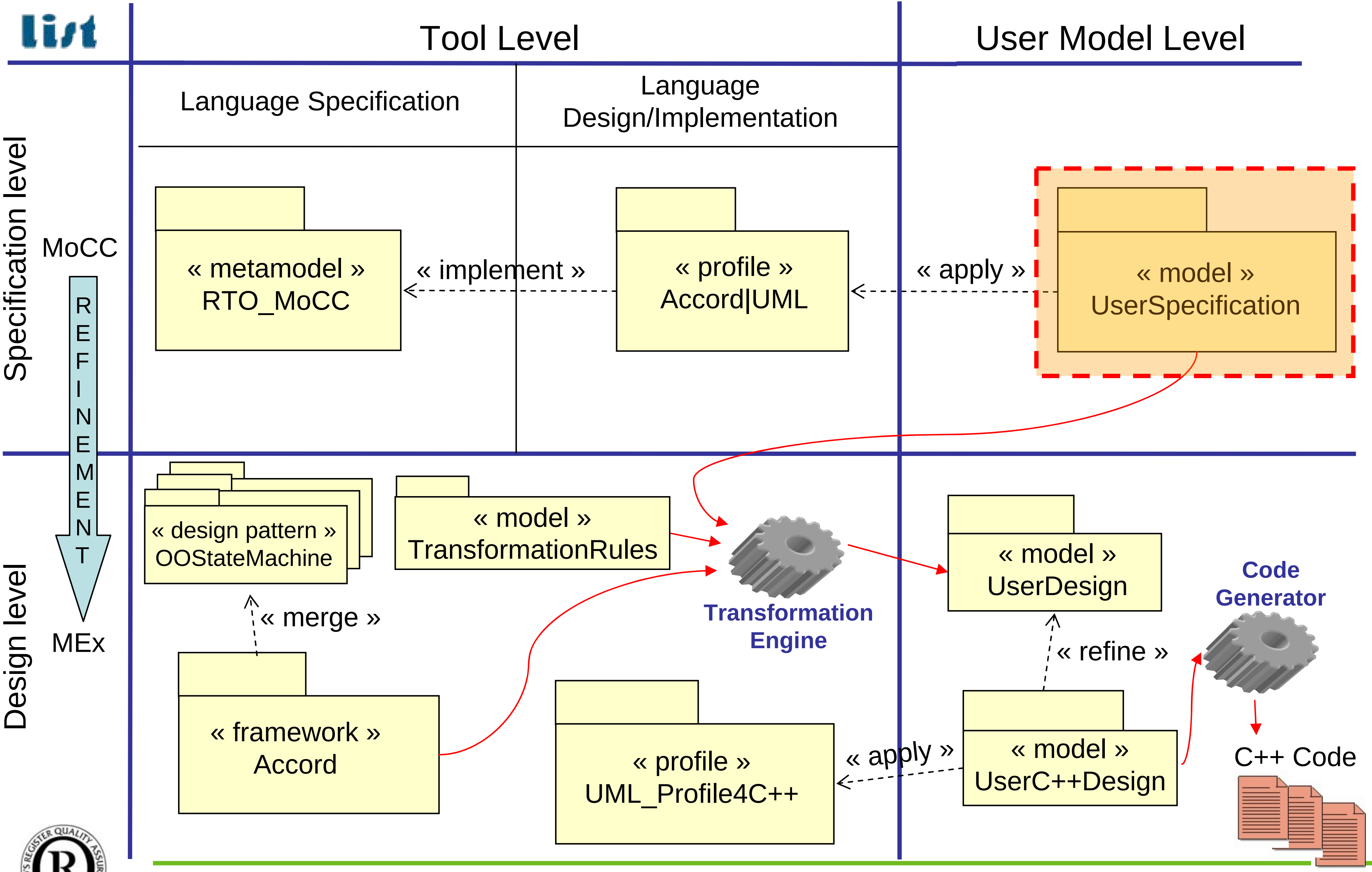
« profile »
Accord|UML

Excerpt…

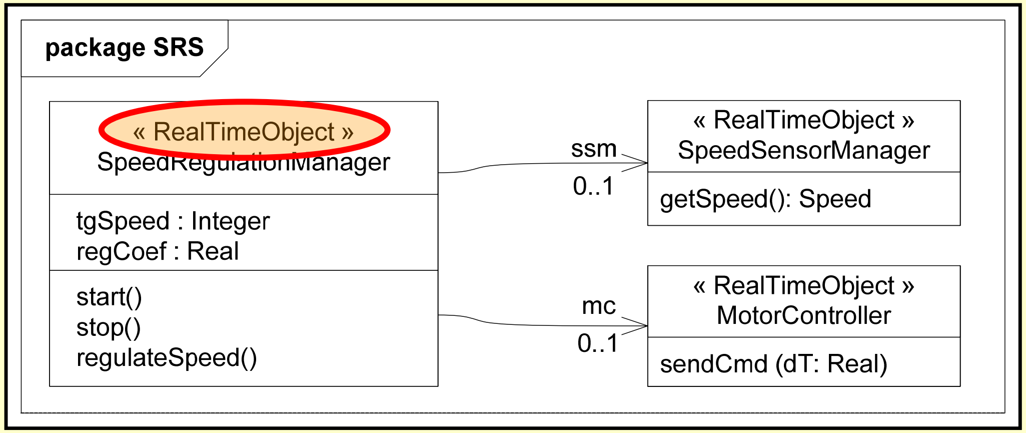


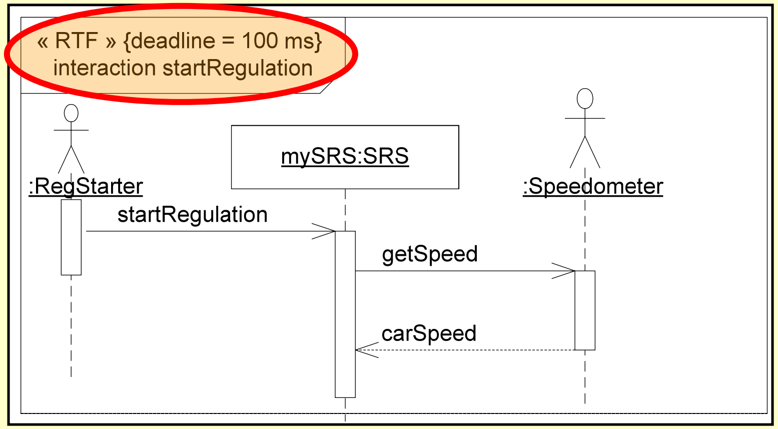
- **Semantic variation points fixed**
  - ✓ Explicitly through stereotype definitions:
    - → Ex1. Scheduling policy:
      - » Messages stored in a mail box (i.e. messages have an associated time stamp)
      - » Messages dequeud according to an EDF policy
    - → Ex2. Concurrency policy:
      - » <<Write>> operations (modify object's state) are executed in mutual exclusion
  - ✓ Implicitly through modeling constraints:
    - → Ex3. No actions on entry or exit of a state
    - → Ex4. No parallel states
- **Introduces real-time oriented features**
  - ✓ Deadlines, periods,…

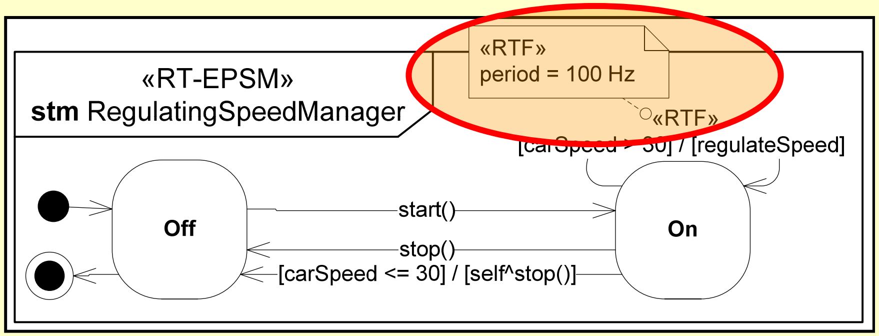
MoCC and MEx within Accord|$_{UML}$

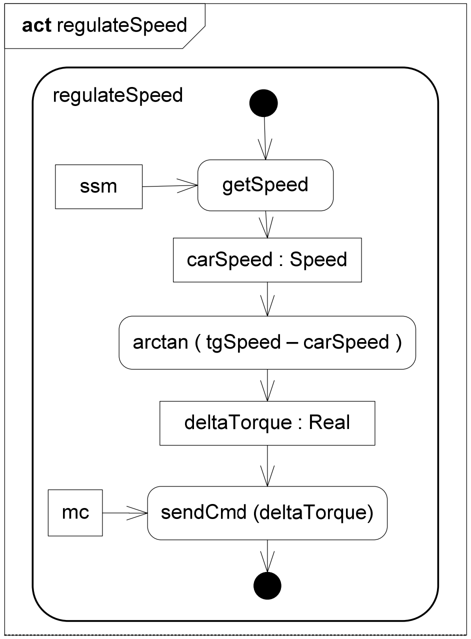


« model » UserApplicationDAM





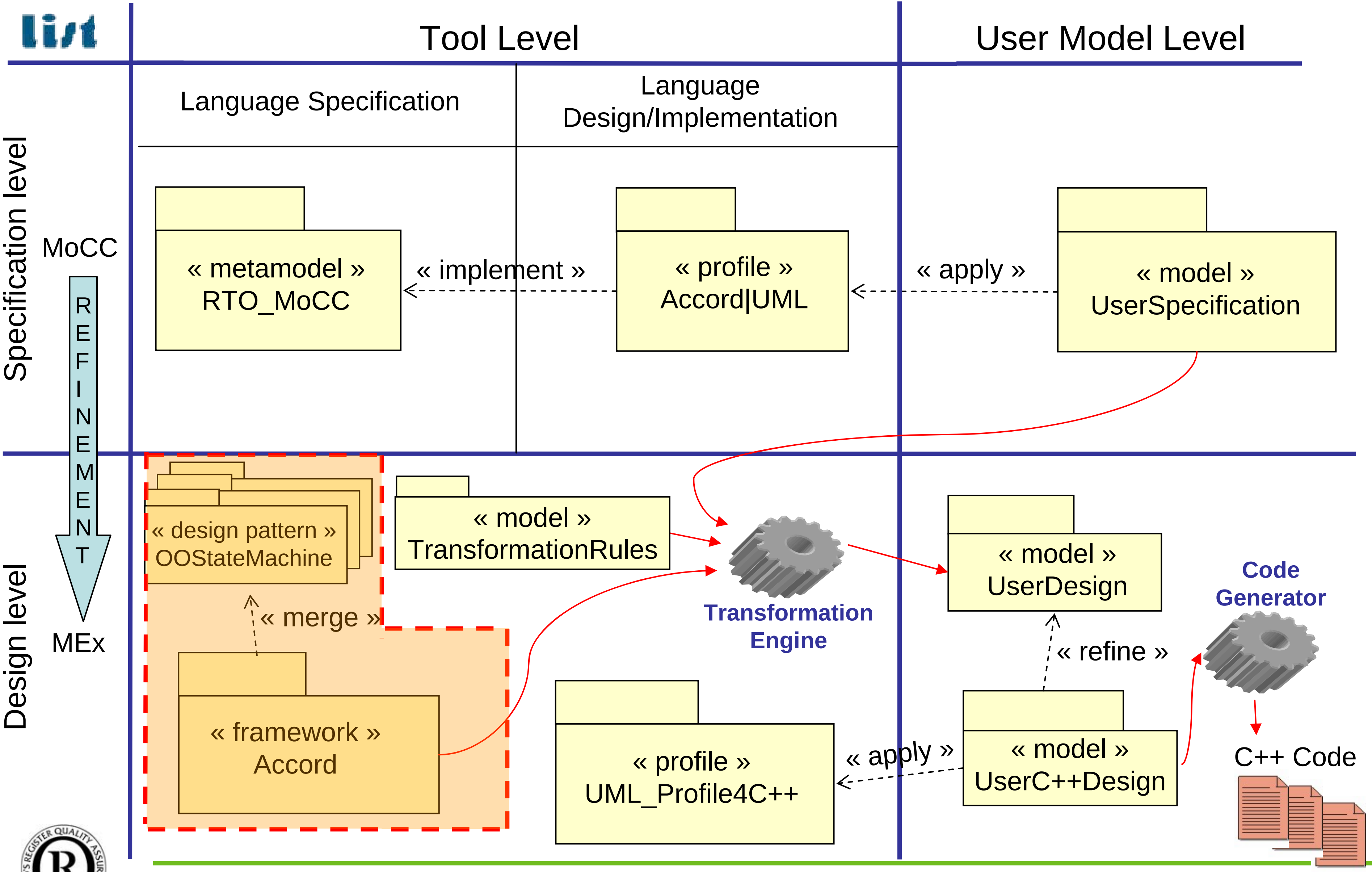

«RT-EPSM»
stm RegulatingSpeedManager
«RTF»
period = 100 Hz
«RTF»
[carSpeed > 30] / [regulateSpeed]
Off
On
start()
stop()
[carSpeed <= 30] / [self^stop()]



« textualView »
act regulateSpeed

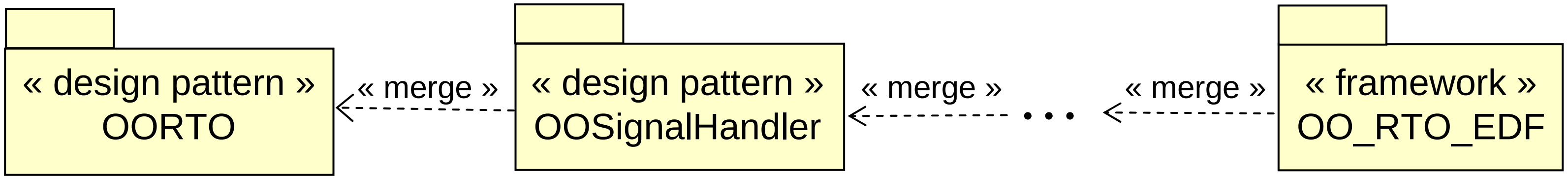
```
// variables declaration
Speed carSpeed
Real deltaTorque
// method body
carSpeed = ssm.getSpeed
deltaTorque = arctan (tgSpeed – carSpeed)
mc.sendCmd (deltaTorque)
```

MoCC and MEx within Accord|$_{UML}$



Focus on the Accord execution frameworks

- **Design patterns**
  - ✓ Behavioral / Structural reference solution for a particular design/implementation issue
  - ✓ Can be « technology » oriented (e.g. Object oriented, component oriented or C++)
- **The Accord model execution frameworks**
  - ✓ Support for execution of application model annotated with Accord MoCC
  - ✓ Refinement of Accord MoCC meta-model for design/implemention purpose
  - ✓ Build by incremental merges ("composition") of specific design patterns
  - ✓ Target of model transformations and code generation for final implementation
- **Two views of the execution framework**
  - ✓ Object oriented view



« design pattern » OORTO

« merge »

« design pattern » OOSignalHandler

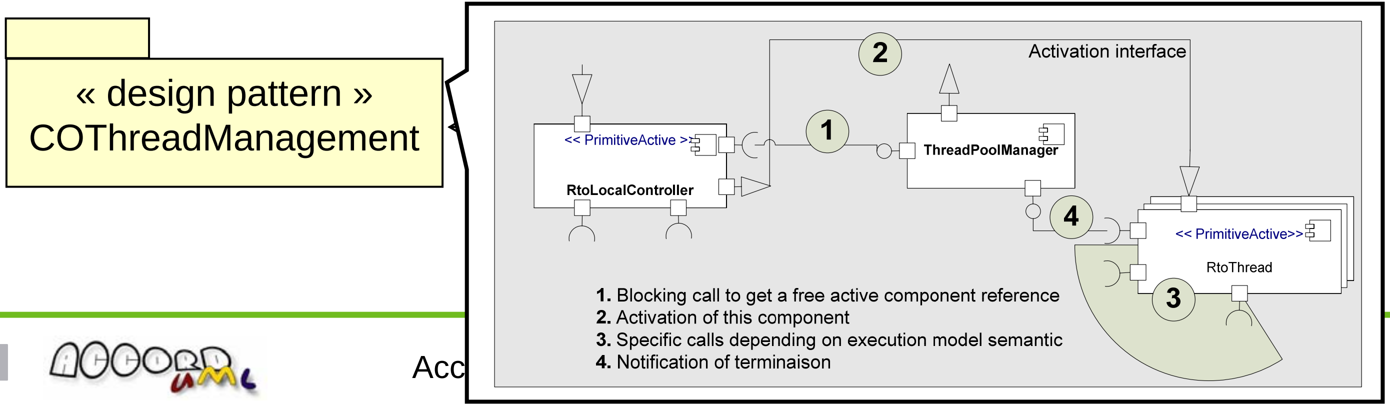
« merge »

…

« merge »

« framework » OO_RTO_EDF

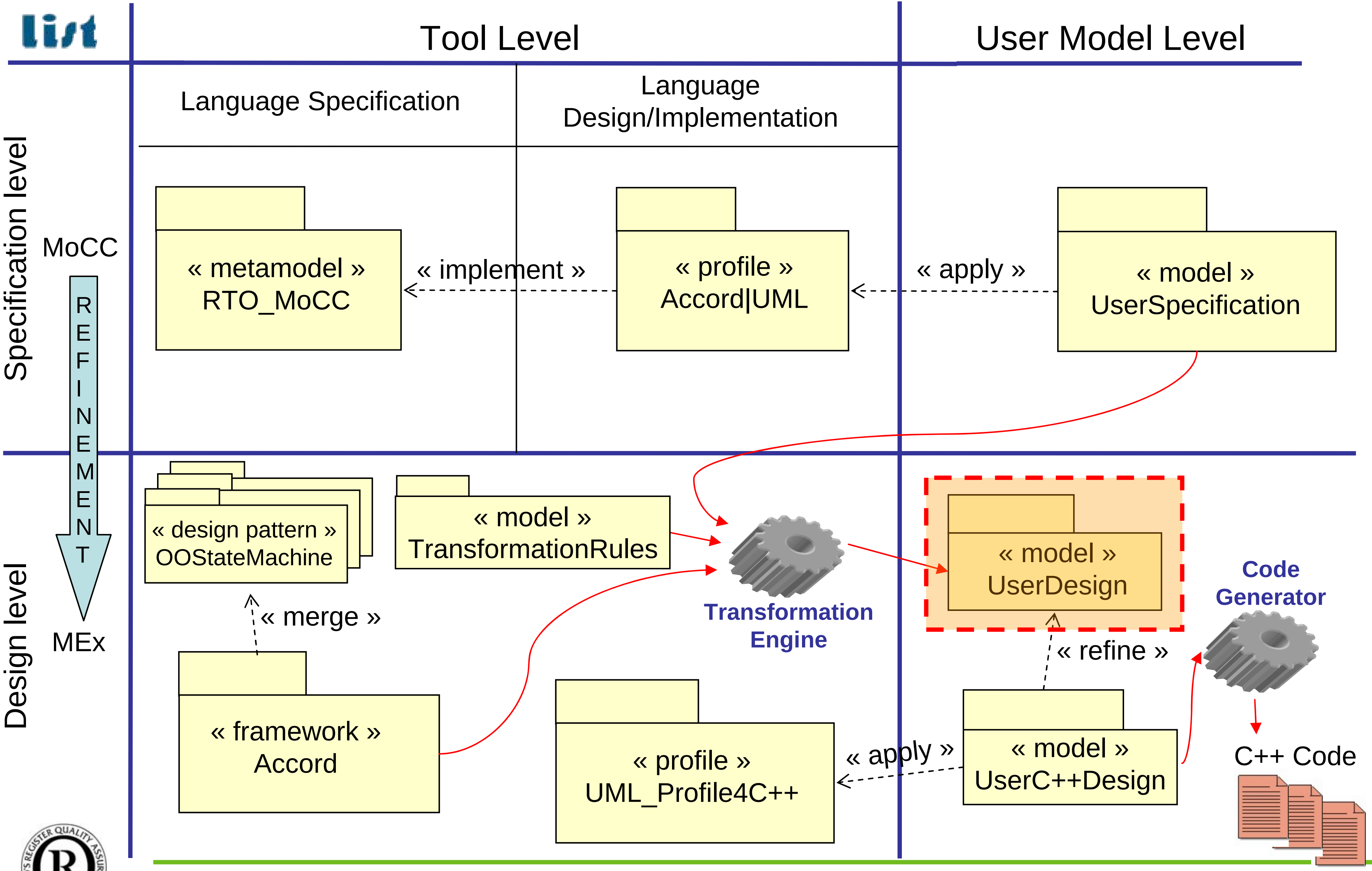
  - ✓ Component oriented view



« design pattern » COThreadManagement

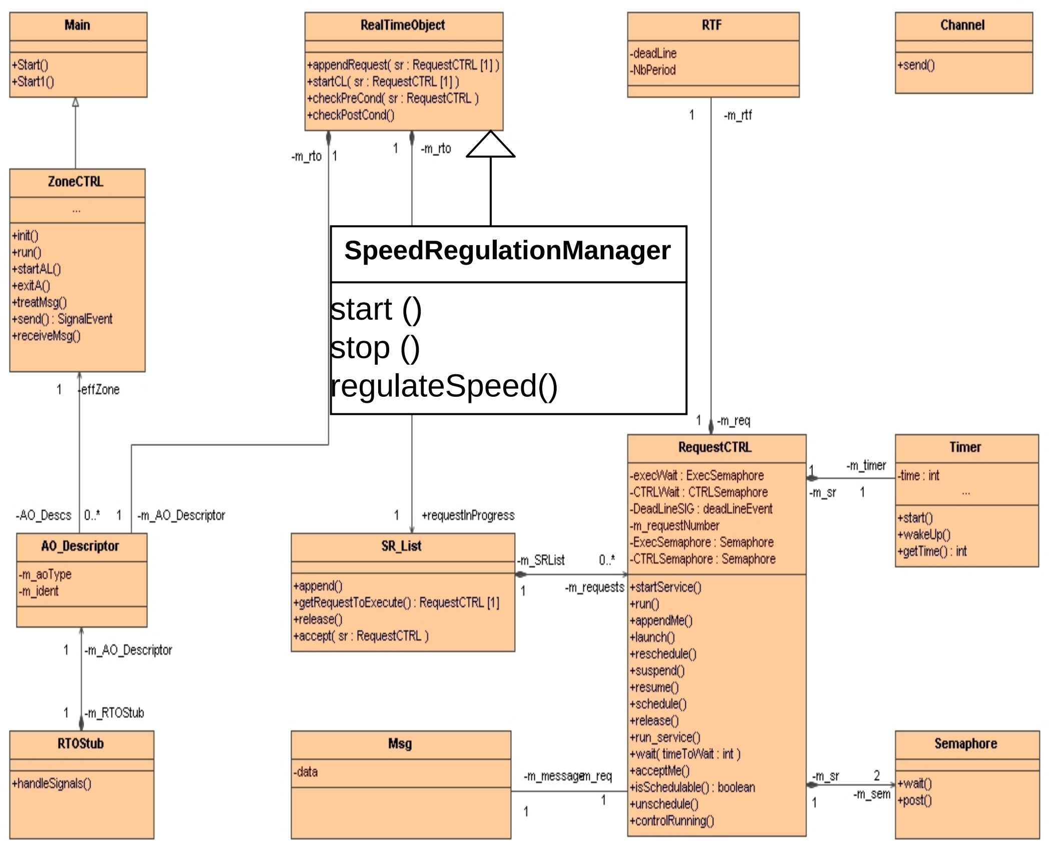
1. Blocking call to get a free active component reference
2. Activation of this component
3. Specific calls depending on execution model semantic
4. Notification of terminaison

MoCC and MEx within Accord|$_{UML}$



« model »
TargetIndependantPrM

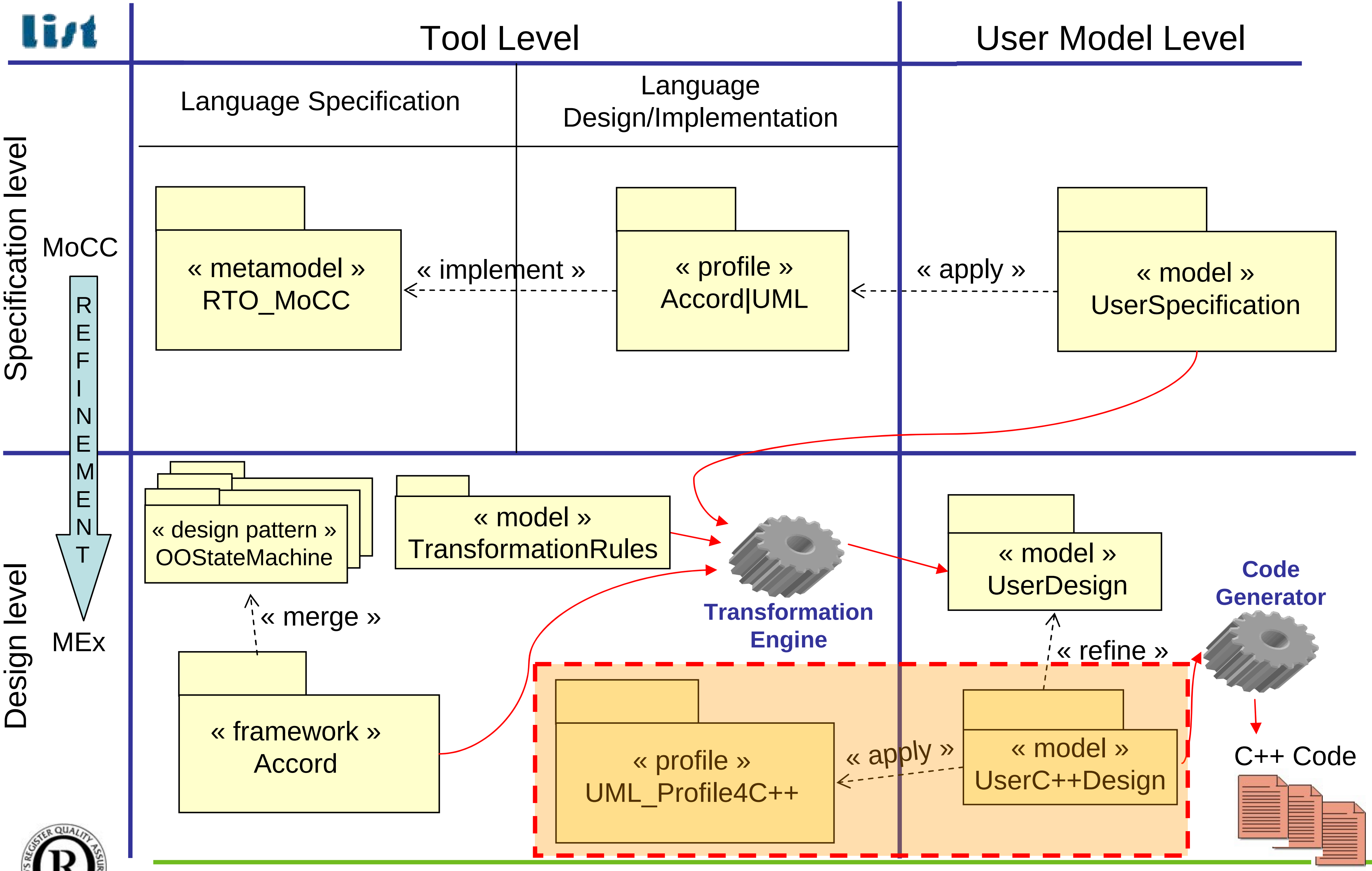
Excerpt…



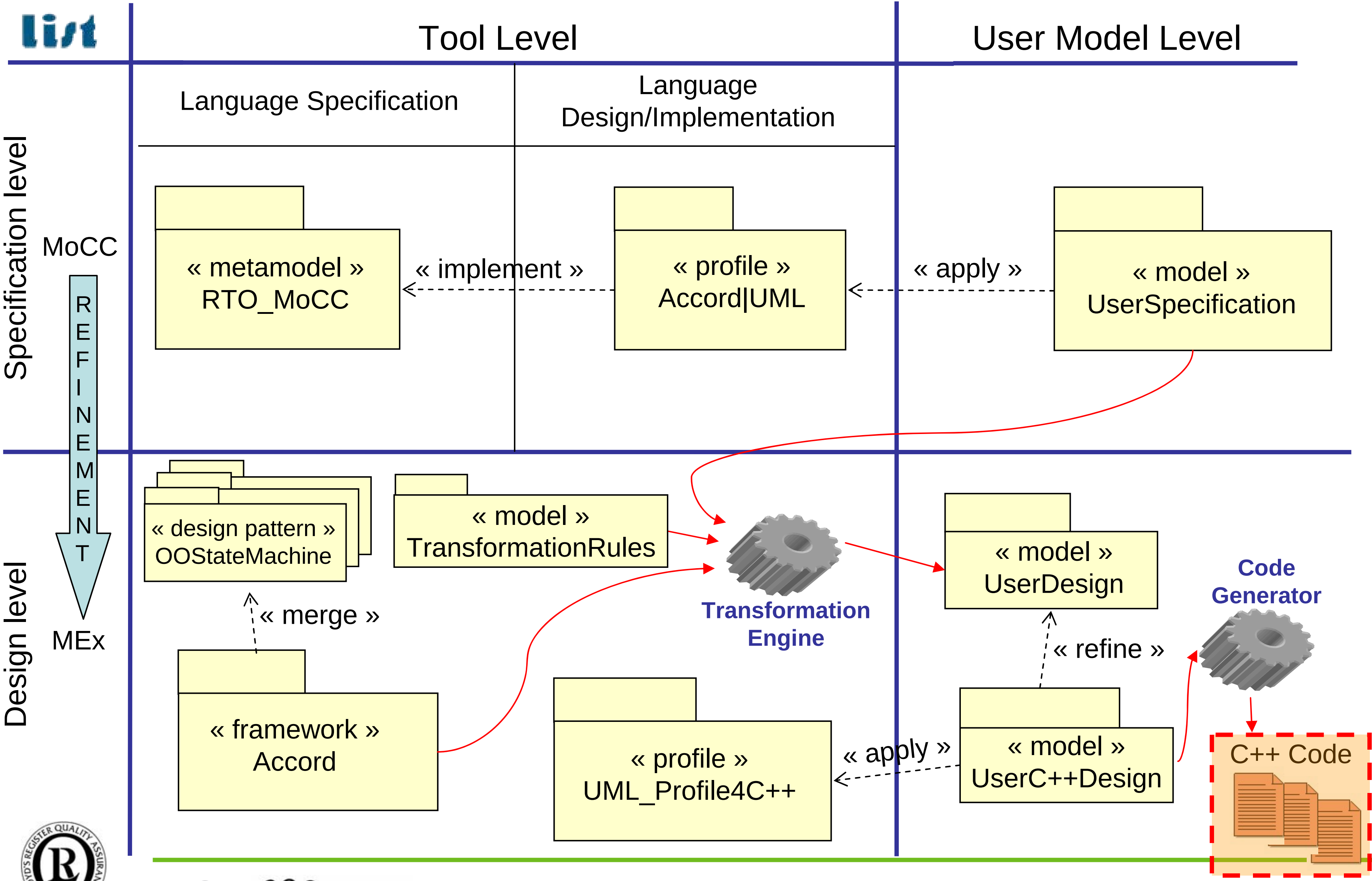
- **Rule examples:**
  - ✓ For each « RealTimeObject » of the DAM, a class with the name is generated.
  - ✓ This class extends the RealTimeObject class of the framework
  - ✓ Method bodies are adapted according to framework specificities

Ex: Transformation guided by an object oriented framework

MoCC and MEx within Accord|$_{\text{UML}}$



MoCC and MEx within Accord|$_{UML}$



« code »
TargetSpecificCode

Excerpt…

```
/************************************************************
 * Code Generated by Accord C++
 * CEA-List
 ************************************************************/
#ifndef SYSTEM_SPEEDREGULATOR_H
#define SYSTEM_SPEEDREGULATOR_H

/************************************************************
              SpeedRegulator class header
 ************************************************************/


/* Owner package header include */
#include <System/Pkg_System.h>

/* Structural includes (inheritance, dependencies...       */
#include <ACCORD_Lib/A_Rbox/UpdateRboxes.hxx>
#include <ACCORD_Lib/A_ActiveObject/SRwithoutDC.hxx>
#include <ACCORD_Lib/A_ActiveObject/RTO_stub.hxx>
#include <ACCORD_Lib/A_ActiveObject/SRwithDC.hxx>

class UpdateRboxes;
class SRwithoutDC;
class RTO_stub;
class SRwithDC;

#include <ACCORD_Lib/A_ActiveObject/SR_Pool.hxx>

class SpeedRegulator;
/* Package type definitions                          */
typedef SetOf< SpeedRegulator*, SpeedRegulator* >
setOfpSpeedRegulator ;

…
```

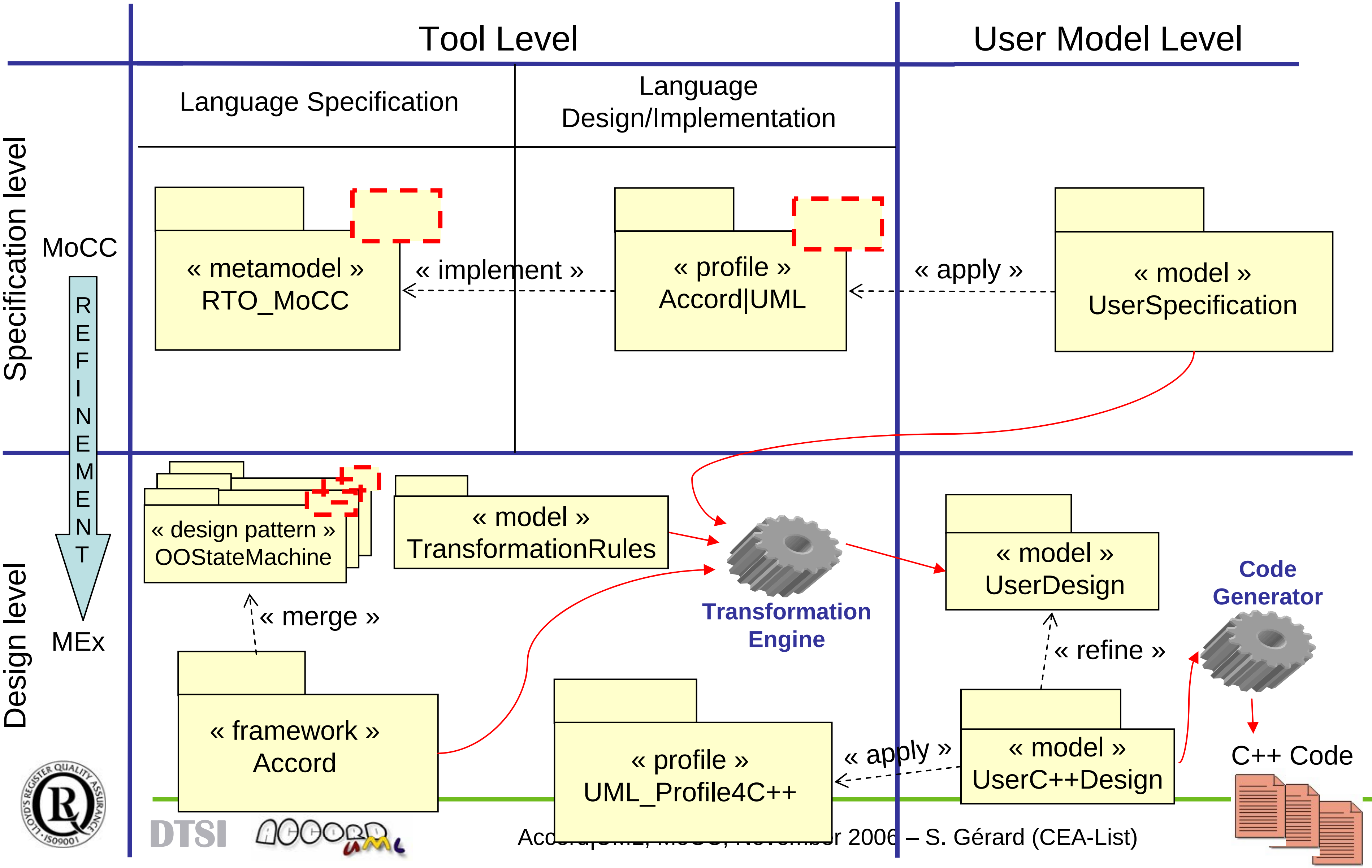
Header of SpeedRegulator

```
/************************************************************
 * Code Generated by Accord C++
 * CEA-List
 ************************************************************/
#define SYSTEM_SPEEDREGULATOR_BODY

/************************************************************
SpeedRegulator class body
 ************************************************************/

/* Header include */
#include <System/SpeedRegulator.h>

/* Include from CppInclude declaration */
#include <ACCORD_Lib/A_Messaging/Signal.hxx>

setOfSpeedRegulator SpeedRegulator::m_instances;

/**
 *
 * @param aRTF_Value
 */
void SpeedRegulator::startRegulation (const RTF& aRTF_Value )  {
   int effZ = m_AO_descriptor.getEffZone ();
   if (effZ != ProDesc::getZoneNumber ()) {


         // Inter-zone request
        ToChannel::send (msg, effZ);
   }
   else {
       // Intra-zone request
      ServiceRequest *sr =
      SR_Pool::getInstance (m_AO_descriptor, NULL);
      sr->getMessageRef () = msg;
      sr->appendMe ();
…
```

Body of SpeedRegulator

Ongoing work: make generic the approach

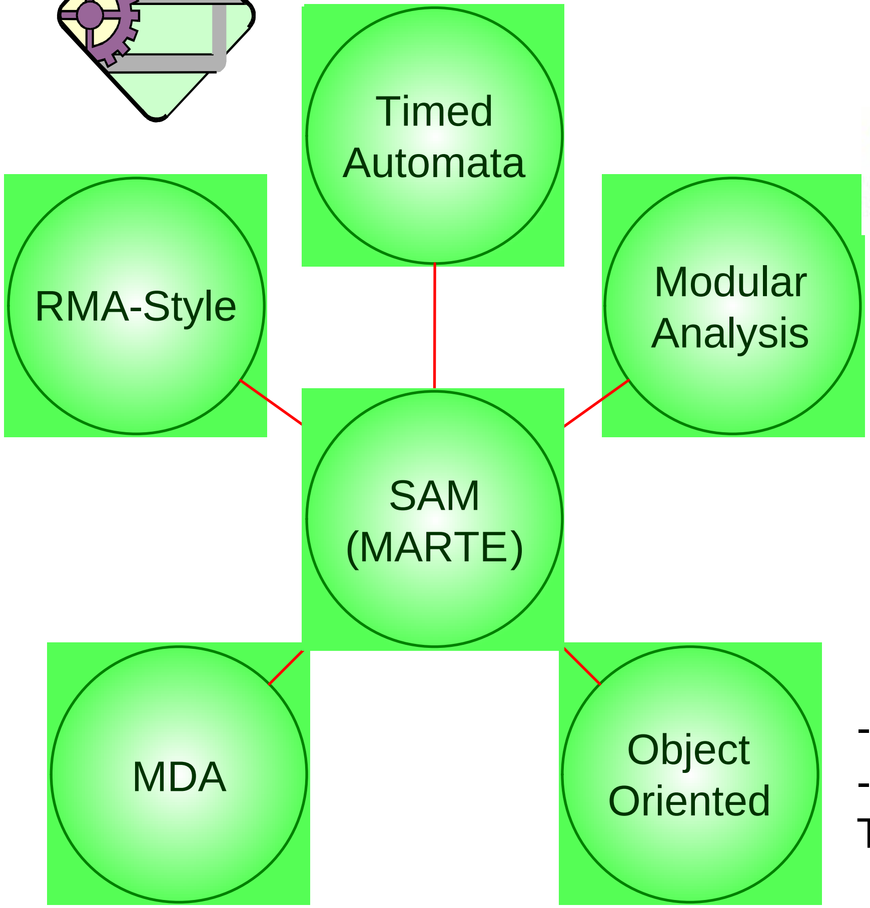
▪ Support for a “template like” mechanism



Next Steps

- **Support for a “template like” mechanism**
- **Semantics formal definition**
  - ✓ System@tic::Usine Logicielle project
    - → Open Dev Factory sub project
      - » PC-xUML Task
        - Define a formal framework dedicated to MoCC spec&design
  - ✓ Bridge to formal verification tools for test generation
    - → Ex: Agatha (Symbolic execution for test generation)
  - ✓ Semantic validation of successive model transformations
    - → From specification…
    - → … to Implementation (i.e. application of design patterns)
- **Unambiguous support for heterogenous MoCCs interoperability**
- **Bridge with Schedulability Analysis tools**

Integrating different RTS models



- Timed automata with tasks

TIMES

AGATHA

- AEIOLTS

- Classic RMA
- Extended RMA
- Holistic Approach

RAPID RMA™

MAST

SYMTA/S 1.0

© Copyright 2006, Symtavision GmbH, Germany All rights reserved!

- Compositional Analysis

- PIM, PSM, PDM

- Active Object Semantic
- Event Priorities vs. Thread Priorities