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 need 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\textsubscript{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 multicast
    - 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|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

![Diagram of Active Object statemachine]

- Semantic issues:
  - Order for dequeing messages (i.e. scheduling policy)?
  - Concurrent modification of an attribute (i.e. internal concurrency management)?
  - When does a message is considered to be handled (i.e granularity of a RTC step) ?
  - and many more…
MoCC of Active Object behaved 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

```plaintext
myActiveObject
att1
att2
opA ()
opB ()
signalA ()
signalB ()
signalC ()
```

---

**Diagram:**
- **Initialize Object**
- **Wait for Event**
- **Handle Event**
- **Terminate Object**

**States:**
- `s1`
- `s2`

**Events:**
- `signalA()` with guard `att1 > 50`
- `signalB()` with guard `att2 <= 30`
- `signalC()`
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 AccordUML methodological framework

- **Requirements Model**
  - Use cases
  - Scenarios

- **Logical Model**
  - Structure
  - Interactions
  - Behaviour

- **Prototype Model**
  - Components
  - Activities

- **Schedule & Performance Model**
  - Workload
  - Behaviour
  - Platform

- **Platform Model**
  - Hardware
  - Software

** <<profile>>**

**Application**

**SAM & PAM**

**HEPM & SEPM**

WCET Calculation
User point of view:

*an object encapsulating data & processing*
User point of view: *an object with its own processing resources*
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|UML

Tool Level
- Language Specification
- Language Design/Implementation
  - « metamodel » RTO_MoCC
  - « implement » Accord|UML
  - « profile » Accord|UML
  - « apply » UserSpecification

User Model Level
- « model » UserSpecification

Specification level
- MoCC
  - « metamodel » RTO_MoCC
- MEx
  - « design pattern » OOSTateMachine
  - « merge » Accord
- Accord|UML
  - « model » TransformationRules
  - « refine » UserDesign
  - « implement » UserC++Design
  - « apply » UserC++Design

Design level
- Accord|UML
- Code Generator
  - C++ Code

Transformation Engine
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 deqeued according to an EDF policy
    - Ex2. Concurrency policy:
      - \(<\text{Write}>\) 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,…
<table>
<thead>
<tr>
<th>Specification level</th>
<th>Tool Level</th>
<th>User Model Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCC</td>
<td></td>
<td>« model » UserSpecification</td>
</tr>
<tr>
<td>REFINEMENT</td>
<td></td>
<td>« implement » Accord</td>
</tr>
<tr>
<td>Design level</td>
<td></td>
<td>« apply » UserC++Design</td>
</tr>
<tr>
<td>MEx</td>
<td>« metamodel » RTO_MoCC</td>
<td>« model » UserDesign</td>
</tr>
<tr>
<td></td>
<td>« design pattern » OOSM</td>
<td>« refine »</td>
</tr>
<tr>
<td></td>
<td>« framework » Accord</td>
<td>« merge »</td>
</tr>
<tr>
<td></td>
<td>« model » TransformationRules</td>
<td></td>
</tr>
<tr>
<td></td>
<td>« profile » Accord</td>
<td>UML</td>
</tr>
<tr>
<td></td>
<td>« profile » UML_Profile4C++</td>
<td></td>
</tr>
<tr>
<td></td>
<td>« model » UserC++Design</td>
<td></td>
</tr>
</tbody>
</table>

Accord|UML, MoCC, November 2006 – S. Gérard (CEA-List)
package SRS

« RealTimeObject »
SpeedRegulationManager

ssm

0.1

tgSpeed : Integer
regCoef : Real

start()
stop()
regulateSpeed()

« RealTimeObject »
SpeedSensorManager

getSpeed(): Speed

« RealTimeObject »
MotorController

sendCmd (dT: Real)

« RT-EPSM »

stm RegulatingSpeedManager

Off

On

stm

RegulatingSpeedManager

« RTF »

period = 100 Hz

« RTF »

« RTF » (deadline = 100 ms)

interaction startRegulation

Vreg

Vacc

Vreg + GainV

[CarSpeed <= 30] / [self.stop()]

Vacc

CarSpeed

// variables declaration
Speed carSpeed
Real deltaTorque

// method body
carSpeed = ssm.getSpeed
deltaTorque = arctan (tgSpeed – carSpeed)
mc.sendCmd (deltaTorque)

regulateSpeed

ssm
getSpeed

carSpeed : Speed

arcctan (tgSpeed—carSpeed)
deltaTorque : Real

mc

sendCmd (deltaTorque)
MoCC and MEx within Accord|UML

Tool Level

Language Specification
- « metamodel » RTO_MoCC
- « design pattern » OOStateMachine
- « framework » Accord
- « model » Accord|UML
- « model » UML_Profile4C++
- « model » UserC++Design
- « model » UserDesign

Language Design/Implementation
- « profile » Accord|UML
- « model » UserSpecification

User Model Level

Specification level

MoCC
- « implement »

Design level

MEx
- « merge »

Transformation Engine
- « refine »
- « apply »

Code Generator
- C++ Code
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/implementation 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
  - Component oriented view
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\textsubscript{UML}

<table>
<thead>
<tr>
<th>Specification level</th>
<th>Design level</th>
<th>Tool Level</th>
<th>User Model Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCC</td>
<td>RTO_MoCC</td>
<td>« implement »</td>
<td>« apply »</td>
</tr>
<tr>
<td>R_E_F_I_N_E_M_T</td>
<td>Accord_UML</td>
<td>« model » UserSpecification</td>
<td></td>
</tr>
<tr>
<td>« design pattern » OOSTateMachine</td>
<td>TransformationRules</td>
<td>« model » UserDesign</td>
<td></td>
</tr>
<tr>
<td>« framework » Accord</td>
<td>« profile » UML_Profile4C++</td>
<td>« apply »</td>
<td></td>
</tr>
<tr>
<td>« model » UserC++Design</td>
<td></td>
<td>« refine »</td>
<td></td>
</tr>
</tbody>
</table>

Transformation Engine

Code Generator

C++ Code
MoCC and MEx within Accord|UML

Tool Level

Language Specification

- « metamodel » RTO_MoCC

Language Design/Implementation

- « profile » Accord|UML

User Model Level

- « model » UserSpecification

Specification level

MoCC

- « framework » Accord

MEx

- « design pattern » OOStateMachine

Design level

« merge »

Transformation

- « model » TransformationRules

- Transformation Engine

- « refine »

User level

- « implement »

- « apply »

C++ Code

- « merge »

Code Generator

« model » UserDesign

- « apply »

« model » UserC++Design

« profile » UML_Profile4C++

Accord|UML, MoCC, November 2006 – S. Gérard (CEA-List)
Excerpt...

```c++
#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

```c++
#endif // SYSTEM_SPEEDREGULATOR_H
```
Ongoing work: make generic the approach

- Support for a “template like” mechanism

<table>
<thead>
<tr>
<th>Tool Level</th>
<th>User Model Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Specification</td>
<td>Language Design/Implementation</td>
</tr>
<tr>
<td><strong>MoCC</strong></td>
<td><strong>UserSpecification</strong></td>
</tr>
<tr>
<td>« metamodel » RTO_MoCC</td>
<td>« apply »</td>
</tr>
<tr>
<td>« design pattern » OOStateMachine</td>
<td>« implement »</td>
</tr>
<tr>
<td>« model » Accord</td>
<td>« profile » Accord</td>
</tr>
<tr>
<td>« framework » Accord</td>
<td>« apply »</td>
</tr>
<tr>
<td>« model » TransformationRules</td>
<td>« model » UserSpecification</td>
</tr>
<tr>
<td>« refine »</td>
<td></td>
</tr>
<tr>
<td>« merge »</td>
<td></td>
</tr>
<tr>
<td>Transformation Engine</td>
<td></td>
</tr>
</tbody>
</table>

- Code Generator
- C++ Code

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
- Classic RMA
- Extended RMA
- Holistic Approach
- Compositional Analysis
- Active Object Semantic
- Event Priorities vs. Thread Priorities