MARTE based methodology and diagram support

Andrey Sadovykh, SOFTEAM
Outlines

• MADES context
• Motivation
  – MARTE presentation
  – Challenges
• Solution in progress
  – Methodology
  – MADES language and diagrams
• Conclusions
MADES context

- Scientific Objective 1: Development of **modelling languages**, and dedicated tools, focusing on avionic and surveillance systems, based on existing generic RTES modelling languages;

- Outreach Objective 4: Liaison with OMG to influence **standardization** of RTES modelling approaches.
MADES requirements

• System engineering
  – Requirements modeling
  – System design
  – Refinements

• Co-design
  – Software design
  – Hardware design (including SoC)
  – Allocation
  – Timing and Scheduling
  – Behavior

• Verification, Simulation and Synthesis
MOTIVATION
State of the Art

• **SysML**
  – First OMG standard for systems engineering
  – Requirements, blocks, ports, parameters
  – Non Functional Properties cannot be expressed: Latency, Throughput, etc….

• **AADL**
  – Originated from avionic domain and standardized by SAE
  – Process and Threads, Communicates via ports, calls and shared memory
  – Verification and validation tools

• **AUTOSAR**
  – Standardized for automotive industry
  – Standardized components, APIs. Timing modelling
  – Too domain specific

• **UML for SoC, UML for SystemC, etc…**
• **UML profile for SPT, ACOTRIS, etc…**
MARTE

• UML Profile for Modeling and Analysis of Real-Time and Embedded systems
• Standardized by OMG on 02/11/2009

• Goals:
  – Providing a common way of modeling both hardware and software aspects of a RTES.
  – Enabling interoperability between development tools used for specification, design, verification, code generation, etc.
  – Fostering quantitative predictions regarding real-time and embedded features of systems taking into account both hardware and software
MARTE overview

- Core Elements
- Non Functional Properties (NFPs)
- Time Modeling
- Generic Resource Modeling (GRM)
- Allocation Modeling
- Generic Component Modeling (GCM)
- High Level Application Modeling (HLAM)
- Detailed Resource Modeling (DRM)
  - Hardware Resource Modeling (HRM)
  - Software Resource Modeling (SRM)
- Generic Quantitative Analysis Modeling (GQAM)
- Schedulability Analysis Modeling (SAM)
- Performance Analysis Modeling (PAM)
- Value Specification Language (VSL)
- Clock Handling facilities
- Repetitive Structure Modeling (RSM)
- MARTE Library
Issues / Challenges

• MARTE is designed to enable flexibility
  – Too many concepts
  – Same concept can be applied to different UML elements (classes, nodes, ports, attributes)
  – Same design may be modeled in different ways using MARTE concepts from different profiles and different UML elements
  – MARTE tools are too generic
  – Lack of guidelines and examples

• MARTE needs tailoring
  – Dedicated methodologies
  – Dedicated tools
MADES SOLUTION
(IN PROGRESS)
MADES modeling language

• Combination of SysML, MARTE and UML
  – High-level specification
  – Detailed design of RTES

• Requirement: **Minimum viable subset**
  – Avoid ambiguity
  – Reinforce formality for V&V and synthesis
  – Better user guidelines

• Approach:
  – Analysis of examples
  – Building methodology
  – Creating dedicated tool support
MADES methodology

- Generic methodology for co-design
### Dedicated diagrams

| High Level specification | **Main:** SysML Requirement, High-level structure, and UML Use cases  
|                          | Behaviour: UML Activity, State Machine, Sequence and Interaction  
|                          | Overview  
|                          | Refinement with: MARTE Time, Hardware, Software and NFP |
| Partitioning, Allocation | **Main:** MARTE Software, Hardware and Allocation  
|                          | Behaviour: UML Activity, State Machine, Sequence and Interaction  
|                          | Overview  
|                          | Refinement with: MARTE Time and NFP |
| Scheduling               | **Main:** MARTE Time  
|                          | Activities: Refinement |
| Synthesis                |
High Level specification

- Capture the requirements.
  - Users requirements
  - Functional and non-functional
- Functional architecture.
  - Specify high level architecture.
  - Relations and communication between high level entities.
- Specify system behavior at high level.
Requirement Capture

- **SysML Requirements**
  - Concepts used: requirements, and their relationships - derive, satisfy, verify, refine, etc.

- **UML Use cases**
  - Concepts used: system, actors, use cases and their relations (extension, include, relation) between use cases and actors.
Control Strategy

Control Level Strategy
The level must be maintained in 70% of the tank capacity through the level and the outflow control the fluid.

Tank Capacity
The tank capacity is unknown but the level device sensing the tank filing amount.

Outflow Device
Control Outflow Strategy
The outflow control is maintained by the positioner device that control the flow fluid amount.

Positioner Device
Use case diagram

Actor

Uses Case

- Send messages
- Receive messages
- Allignement phase

<<include>>

- Prepare data for messages
- Compute status
- Store output data

<<extend>>
Functional Architecture

• Based on UML composite diagram.

• Describes the system as a set of modular “blocks”.

• Captures the block definitions, their relationships and communication means.

• Mix between SysML block definition diagram and MARTE GCM.

• Concepts used: block, interface, port, flow, connector, provided/required, operation.
System behavior

- Using UML2 Behavioral diagrams:
  - Sequence Diagram
  - State Machines Diagram
  - Activity Diagram
  - Internal Overview Diagrams.
With Sequence Diagrams

User:

<<HwComputingResource, HwComponent>>
  terminal: Terminal

<<HwBus, HwComponent>>
  pci-bus: PCI-Bus

<<HwComputingResource, HwComponent>>
  gpu: GraphicalProcessingUnit

send input

received

par

[]

DisplayMessage

[]

ExecuteTask
With State Machine Diagram

- Executing
  - Do / Executing
    - Sleep
      - Do / Sleep
    - Idle
      - Do / Idle
With Activity Diagrams

<<GaWorkloadEvent>> CameraEvent

<<GaScenario>>
CameraDistanceCalculation
Allocation, Partitioning

• Definition of hardware architecture.

• Definition of software architecture.

• Mapping (allocation) between set of software elements and hardware architecture.
Hardware architecture diagram

- Based on UML component and composite structure diagrams.
- Subset MARTE hardware profile.
- Concepts used: component, interface, part, port, connector, interface realization, usage.
- MARTE library for Hardware Types
Hardware architecture diagram
Software architecture diagram

- Based on class diagram.

- Subset of MARTE software profile.

- **Concepts used:** class, interface, package, property, operation, association, generalization, realization, usage, etc.
• Based on MARTE Allocation

• Map elements onto the available architecture.

• Temporal and spatial allocation.
Allocation diagram
Scheduling

- Definition of timing constraints
- Based on UML2 Behavioral diagrams i.e. Sequence, State Machine, Activity, Internal Overview Diagrams.
Time diagram

- Coming from MARTE.
- Depict the time structure.
- **Concepts used:** ClockType, Clock, ClockConstraint elements.
- Starting point of any timing specification.
Scheduling

\{t_{10} - t_{00} \leq 8, ms\}

getLocation()

getLocationData

getFlightPlan()

FlightPlanData

paramUpdated(paramValue)
CONCLUSIONS
Conclusions

- MADESi in progress for development of a modeling language for RTES
  - Combination of SysML, MARTE and UML
  - Supported by methodology and dedicated tools
  - Reinforce formality and user-friendliness

- Implemented in Modelio by SOFTEAM

- Contributes to MARTE Revision Taskforce in OMG
Thank you very much for your attention

Questions?
SYSML OVERVIEW
SysML

• graphical modeling language for specifying, analyzing, designing, and verifying complex systems

• In particular, representations modeling system requirements, behavior, structure, and parametrics.
• Uses Modelio Requirement Analyst module

• Features:
  – requirement analysis,
  – compliant with the SysML standard,
  – traceability management,
  – impact analysis diagrams,
  – dedicated documentation generation, etc.
Activity Diagram

<<AllocateActivityPartition>> LevelDevice

- Read in level_input:integer
  - Control the Level
    - Read Level Value
  - Read Reference Value

<<AllocateActivityPartition>> OutflowDevice

- Read in outflow_input:integer
  - Control the Outflow
    - Read Outflow Value
Parametric Diagram

Weight calculation

Friction: 

DutyCycle: 

BrakingForce: 

\[ f = (tf \cdot bf) \cdot (1 - tl) \]

Distance Equation: 

\[ v = \frac{dx}{dt} \]
Example – the Plant

• An industrial automation experimental unit.

• SysML describes:
  • its structure,
  • its behavior,
  • its requirements
MARTE Presentation

Etienne Brosse
Outlines

• Introduction
  – Historic Background
  – Goals

• MARTE Specification
  – Structure
  – Profiles description

• Next stages
  – Discussion
  – Planning
Historic Background

• Modeling and Analysis of Real-Time and Embedded systems.
• Version 1.0 November 2009.
• Normalization in progress by OMG.
• Successor of SPT (Schedulability, Performance and Time).
• Successor de MARTES (Model-based Approach to Real-Time Embedded Systems development)
Goals

• Provides support during many phases:
  – Specification
    • Non functional properties,
    • Time constraint.
  – Design
    • Hardware and Software resource,
    • Resources allocation.
  – Testing/Validation
    • Performance Analysing
• Describes how
  – Specify a non-functional properties (qualitative or quantitative)
  – Describes relation between them
• MARTE library
  – Primitives Types (Real, Date, etc.)
  – Data Types (Array, IntegerVector, IntegerMatrix, etc.)
  – NFP Type (NFP_Frequency, NFP_Percentage, NFP_Power, etc.)
• Used in profile definition
Non-Functional Properties

Processor

frequency : NFP_Frequency

myProcessor:Processor

frequency=2,6 GHz

<<HwProcessor>>
myProcessor {frequency (2,6 GHz)}
Time

• Two aspects:
  – Logical time
    • After, before
  – Real time.
    • Clock, delay, duration, etc.

• Structure:
  – Set of knowing time + relations.

• Time Access (Clock).

• Principe: associating clock and model element.
• Logical Time:
  – Focus on events and their orders.
• Real time:

```plaintext
<<Clock>>
cc1: Chronometric
resolution = 0.01

<<Clock>>
idealClk: IdealClock

<<Clock>>
cc2: Chronometric
resolution = 0.01

{Clock c is idealClk discretizedBy 0.001;
cc1 isPeriodicOn c period 10;
cc2 isPeriodicOn c period 10;
cc1 hasStability 1E-4;
cc2 hasStability 1E-4;
cc1, cc2 haveOffset [0..5] ms wrt idealClk;}
```
Time

idealClk

10 ms  1 ms

(c)

cc1

cc2

offset of cc1 vs. cc2

cc1.period
Generic Component Model

- Concepts for components modeling (UML2, SysML, etc.).

- Two kind of port:
  - Flow Port
    - Data flow oriented
  - ServerClient Port
    - Message flow.
Generic Resource Modeling

- Introduce the resource concept
- Resource specialized into:
  - StorageResource, ConcurrencyResource,
  - SynchronizationResource, MutualExclusionResource,
  - ComputingResource, DeviceResource,
  - CommunicationMedia, CommunicationEndPoint.

- GRM redefined by
  - Software Resource Modeling (SRM)
  - Hardware Resource Modeling (HRM)
Modeling execution context.

**SRM::SW_Concurrency**
- SwScheduleableResource
- InterruptResource
- MemoryPartition
- Alarm
- EntryPoint
- SwTimerResource

**SRM::SW_Interaction**
- MessageComResource
- NotificationResource
- SharedDataResource
- SwMutualExclusionResource

**SRM::SW_Brokering**
- MemoryBroker
- DeviceBroker
Software Resource Modeling

- Example: Event Notification
Hardware Resource Modeling

• **Logical View**

- **HwComputing**
  - `HwProcessor`, `HwPLD`, `HwASIC`

- **HwStorage**
  - `HwCache`, `HwRAM`, `HwROM`, `HwDrive`
  - `HwMMU`, `HwDMA`

- **HwDevice**
  - `HwDevice`, `HwSupport`
  - `HwI/O`

- **HwCommunication**
  - `HwBridge`
  - `HwArbiter`
  - `HwMedia`, `HwBus`

- **HwTiming**
  - `HwClock`, `HwTimer`
Hardware Resource Modeling

• Physical View
  – Focus on layout (forms, size) and power

```
HwComponent
kind: {Card, Channel, Chip, Port}

HwLayout

HwPower
```

« HwPowerSupply »  « HwCoolingSupply »
Hardware Resource Modeling

```
grid = 4.3
area = 5000mm²
r_conditions = (Temperature; Operating; "\[10°C,60°C\]"

« hwCard »
  smp : SMP

<table>
<thead>
<tr>
<th>« hwChip »</th>
<th>« hwChip »</th>
<th>« hwCard »</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu1 : CPU</td>
<td>cpu3 : CPU</td>
<td>sdras : SDRAM</td>
</tr>
<tr>
<td>position = [1,1], [1,1]</td>
<td>position = [2,2], [1,1]</td>
<td>position = [3,4], [1,1]</td>
</tr>
<tr>
<td>staticConsumption = 5W</td>
<td>staticConsumption = 5W</td>
<td>nbPins = 144</td>
</tr>
</tbody>
</table>

« hwChannel »
  fsb : FSB
  position = [1,4], [2,2]

<table>
<thead>
<tr>
<th>« hwChip »</th>
<th>« hwChip »</th>
<th>« hwChip »</th>
<th>« hwPowerSupply »</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu2 : CPU</td>
<td>cpu4 : CPU</td>
<td>dma : DMA</td>
<td>battery : Battery</td>
</tr>
<tr>
<td>position = [1,1], [3,3]</td>
<td>position = [2,2], [3,3]</td>
<td>position = [3,3], [3,3]</td>
<td></td>
</tr>
<tr>
<td>staticConsumption = 5W</td>
<td>staticConsumption = 5W</td>
<td></td>
<td>capacity = [4,4], [3,3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>weight = 150g</td>
</tr>
</tbody>
</table>
```
Allocating an element a hardware element
Analysis profiles

• 3 profiles dedicated to analyse:
  – Generic Quantitative Analysis Modeling (GQAM)
  – Schedulability Analysis Modeling (SAM)
  – Performance Analysis Modeling (PAM)

• Element Annotation
Discussion

• What’s wrong:
  – Too generic and large
  – Questionable design decisions
    • Too many stereotype non used
    • Massive use of stereotype superposition
    • Clock relations modeling
    • Not same detail level in sub profiles.
  – Hardware layout is out of scope
  – No methodology
    • No examples
    • Too many stereotypes
    • No dedicated diagrams definition.