Modeling Languages for Realtime and Embedded Systems
Requirements and Standards-Based Solutions*

Andre Backofen

* Sébastien Gérard, Huascar Espinoza, François Terrier, Bran Selic
Introduction

- Modern Realtime and Embedded Systems (RTES)
  - Complex nature
  - More sophisticated
  - More dependable

- Traditional code-based approaches
  - Reaching the limits of effectiveness

- Use of abstraction to cope with complexity
  - Model-based design

ESC\(^1\)

~ 70 electronic control units
> 100 million instructions

\(^1\) Electronic Stability Control

Actor
Sensor
ESC\(^1\)
Electronic Stability Control
Model-based design

- Incremental design process
  - Based on progressive refinement
  - Performed until the model either
    a) is sufficiently detailed for straightforward implementation
    b) becomes the system
      - E.g. code generation

- Correct-by-construction
  - Automated transformations
  - Computer-based analyses of correctness
Realtime and Embedded Systems domain

Just Software on Small Computers?

- RTES\(^1\) interact with the real world
  - E.g. time is an important issue

- System is compositions of different interrelated parts
  - Interplay of multiple disciplines
    - Parts designed by different teams
  - Pre-designed parts
    - E.g. delivered by automotive suppliers

\(^1\) Realtime and Embedded System
Outline

- Introduction
- **Requirements for Modeling Languages for RTES**
  - Allocation
  - Resource
  - Refinement
  - Time
  - Quantitative Aspects
  - Qualitative Concerns
- Standards-Based Solutions
- Conclusion
Allocation

- The design of RTES\(^1\) often follows the Y-Chart scheme
  - Specify the application model
  - Specify the resource platform model

- Use a third model (allocation)
  - Specify how to combine the models to provide the full system model

- Add non-functional characteristics to the allocation
  - E.g. worst-case execution time

\(^1\) Realtime and Embedded System
Resource

- RTES interact with the real world
  - Constrained by the real world
    - Typically resource limited
    - Need precise modeling of such resources
    - Perform resource optimizations

- Embedded in underlying hardware/software platforms
  - Generated code must be easily interfaced with several platforms
Refinement

- Support refinement relationships between two model layers
  - Relate the different layer models
  - Conformance verification
  - Derivation of model elements
    - From one level to the next

- Attach non-functional properties to refinement relationships

- Trace and propagate changes up and down the layer hierarchy
Time

Different models of time:

a) Asynchronous / Causal models
   - Ordering of activities
   - Time is modeled as causal dependencies

b) Synchronous / Clocked models
   - Add simultaneity of activities
   - Time is modeled as a discrete set of instants

c) Real / Continuous time models
   - Add physical durations
   - Used for time-related analyses
     • e.g. deadline matches
Quantitative Aspects  
better known as Analysis

- Different analysis techniques
  - E.g. stress or thermal analyses (mechanical)
  - Identify or predict quality attributes of a system

- Annotate models with relevant information
  - Add non-functional characteristics to system model
  - E.g. performance, power consumption

- Ensure: analysis model ≈ system
  - Derive analysis model from system model
    - E.g. automated support
Qualitative Concerns
better known as Concurrent, Computation and Communication

- RTES are closely coupled to the real world
  - The real world is inherently concurrent
    - Specify concurrent entities
    - Support interaction between them

- System is a composition of different interrelated parts
  - Developed using different models of computation and communication
## Requirements for Modeling Languages for RTES

<table>
<thead>
<tr>
<th>Y-Chart scheme</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refinement relationships between layers</td>
<td>Resource</td>
</tr>
<tr>
<td>Concurrent, Computation and Communication</td>
<td>Refinement</td>
</tr>
<tr>
<td></td>
<td>Quantitative Concerns</td>
</tr>
<tr>
<td></td>
<td>Qualitative Concerns</td>
</tr>
<tr>
<td></td>
<td>Time</td>
</tr>
</tbody>
</table>

- **Resources modeling**
- **Analysis**
- **Different models of time**
Outline

- Introduction
- Requirements for Modeling Languages for RTES
- Standards-Based Solutions
  - MARTE
  - SysML
  - AADL
  - AUTOSAR
  - EAST-ADL
- Conclusion
Standards-Based Solutions

- **MARTE**
  - OMG\(^1\) standard since 2009
  - Modeling and analysis of RTES

- **SysML**
  - OMG standard since 2007
  - System Modeling

- **AADL**
  - SAE\(^2\) standard since 2004
  - Modeling and analysis of RTES

- **AUTOSAR**
  - Industry standard since 2005
  - Automotive domain

- **EAST-ADL**
  - Developed by an EU project since 2004
  - Software and System modeling
  - Automotive domain

---

1 Object Management Group
2 Society of Automotive Engineers
Modeling and Analysis of Real-Time and Embedded systems

- Modeling, analysis and simulation concerns are tackled

- Detailed modeling of time as the basis for real-time systems

- UML profile
  - Lightweight extension of the UML
  - Hierarchy of sub-profiles

---

Architecture of MARTE

- MARTE foundations
- MARTE design model
- MARTE analysis model
- MARTE annexes

1 annex – attachment, extra
Refined Architecture of MARTE

Non-Functional Properties

High Level Application Modeling

Hardware Resource Modeling

Value Specification Language

Time Modeling

Allocation Modeling

Generic Quantitative Analysis Modeling

MARTE foundations

- «profile» NFP
- «profile» Time
- «profile» GRM
- «profile» Alloc

MARTE design model

- «profile» GCM
- «profile» HLAM
- «profile» SRM
- «profile» HRM

MARTE analysis model

- «profile» GQAM
- «profile» SAM
- «profile» PAM

MARTE annexes

- «profile» VSL
- «profile» RSM
- «modelLibrary» MARTE_Library
MARTE Example

- Modeling a CruiseControlSystem of a miniature robot
  - CruiseController regulates the speed of the robot

- Infrared sensors cover 360° of the environment
  - ObstacleDetector detects obstacles
  - Stops, if an obstacle is detected

- A Speedometer represents a data container for speed
  - Shared between CruiseController and ObstacleDetector

1 obstacle – barrier, blocking object
Design of a CruiseControlSystem

- CruiseController and ObstacleDetector are active units
  - Running concurrently

- Speedometer is a passive unit
  - Access to the data need to be protected
  - concPolicy: CallConcurrencyKind
    - sequential, guarded or concurrent
Collaboration of CruiseController and ObstacleDetector

- **CruiseController**
  - «TimedConstraint» \( t1[i] - t0[i] < (8, ms) \)
  - «rtFeature» \( \text{getSpeed()} \)
  - «rtFeature» \( \text{speed = noSpeed} \)
  - \( \text{occKind=aperiodic()} \)

- **Speedometer**
  - «rtFeature» \( \text{setSpeed(noSpeed)} \)
  - «rtFeature» \( \text{stop()} \)
  - \( \text{occKind=aperiodic()} \)

- **ObstacleDetector**
  - «rtAction, rtFeature» \( \text{isObscatedDetected()} \)
  - \( \text{priority=1} \)
  - \( \text{occKind=aperiodic()} \)
  - \( \text{relDL=(3, ms)} \)
  - \( \text{tRef=t0} \)
  - \( \text{miss=(1, %, max)} \)
  - \( \text{syncKind=delayedSynchronous} \)

- **noSpeed**
  - \( \text{occKind=aperiodic()} \)
  - \( \text{relDL=(10, ms)} \)
  - \( \text{tRef=t1} \)
  - \( \text{miss=(1, %, max)} \)
  - \( \text{syncKind=synchronous} \)
Does MARTE fulfill the Requirements?

MARTE was designed to cover all Requirements!

<table>
<thead>
<tr>
<th>MARTE foundations</th>
<th>MARTE design model</th>
<th>MARTE analysis model</th>
<th>MARTE annexes</th>
</tr>
</thead>
<tbody>
<tr>
<td>«profile» NFP</td>
<td>«profile» GCM</td>
<td>«profile» SAM</td>
<td>«profile» VSL</td>
</tr>
<tr>
<td>«profile» Time</td>
<td>«profile» HLAM</td>
<td>«profile» PAM</td>
<td>«profile» RSM</td>
</tr>
<tr>
<td>«profile» GRM</td>
<td>«profile» HRM</td>
<td></td>
<td>«modelLibrary» MARTE_Library</td>
</tr>
<tr>
<td>«profile» Alloc</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Allocation**: ✔
- **Resource**: ✔
- **Refinement**: ✔
- **Quantitative Concerns**: ✔
- **Qualitative Concerns**: ✔
- **Time**: ✔

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Supported</th>
<th>Partially supported</th>
<th>Not supported</th>
<th>Not well-defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refinement</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantitative Concerns</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Qualitative Concerns</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- ✔ Supported
- ✗ Not supported
- ? Not well-defined
- **System Modeling Language**
  - Subset of UML with extensions
  - Specify and analyze complex systems
  - Systems may include e.g. hardware, software, information, personnel

- MARTE and SysML are complementary
  - SysML deals with many of the same areas
    - Works at a different abstraction level
  - Syntax and semantic of MARTE is aligned to SysML

- Supports various types of allocation
- Uses *only* UML time constraints
### AADL

**Architecture Analysis and Design Language**
- Domain-specific language from scratch
- Specify and analyze RTES
- Systems may include hardware and software

**MARTE and AADL focus on modeling and analysis of RTES**
- MARTE rendering of AADL exists
- MARTE incorporates experience from the AADL community

**Supports different clocks and time constraints**

<table>
<thead>
<tr>
<th>Allocation</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>✓</td>
</tr>
<tr>
<td>Refinement</td>
<td>?</td>
</tr>
<tr>
<td>Quantitative Concerns</td>
<td>✓ ?</td>
</tr>
<tr>
<td>Qualitative Concerns</td>
<td>✓</td>
</tr>
<tr>
<td>Time</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Supported**

**Not supported**

**Partially supported**

**Not well-defined**
AUTomotive Open System Architecture

- The standard in the automotive domain
- Specify component-based software infrastructure
- Decouple software applications from the underlying hardware platform
- Framework includes Architecture, Application Interfaces and Methodology

- MARTE and AUTOSAR are complementary
- Time not supported
  - Extension TIMMO\(^1\) adds time-support
- Works on a low abstraction level

<table>
<thead>
<tr>
<th>Autosar</th>
<th>Allocation</th>
<th>✔</th>
<th>Resource</th>
<th>✗</th>
<th>Refinement</th>
<th>✗</th>
<th>Quantitative Concerns</th>
<th>✗</th>
<th>Qualitative Concerns</th>
<th>⊂</th>
<th>Time</th>
<th>✗</th>
</tr>
</thead>
</table>

\(^1\) Time Model (Project of the C-Lab and others)
EAST-ADL

- **Electronics Architecture and Software Technology – Architecture Description Language**
  - Combination of UML and natural language
  - Complements AUTOSAR
    - EAST-ADL for system architecture
    - AUTOSAR for implementation in software
  - Possible to describe EAST-ADL-like models with a subset of MARTE concepts
  - Time not yet supported
    - Extension TIMMO adds time-support
  - EAST-ADL includes early analysis

<table>
<thead>
<tr>
<th>EAST-ADL</th>
<th>Allocation</th>
<th>Resource</th>
<th>Refinement</th>
<th>Quantitative Concerns</th>
<th>Qualitative Concerns</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>☞</td>
<td>☒</td>
<td>☑</td>
<td>☐</td>
<td>☐</td>
<td>☒</td>
</tr>
</tbody>
</table>

- Supported
- Not supported
- Partially supported
- Not well-defined
Conclusion

- Complexity of RTES
  - **Model-based design** used to cope with complexity

- Interplay of multiple disciplines
  - common modeling language needed

- Several **standards-based solutions**
  - Popular modeling languages introduced
  - Other languages also possible
    - E.g. Mechatronic UML

<table>
<thead>
<tr>
<th></th>
<th>MARTE</th>
<th>SysML</th>
<th>AADL</th>
<th>Autosar</th>
<th>EAST-ADL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>◄</td>
</tr>
<tr>
<td>Resource</td>
<td>✔</td>
<td>x?</td>
<td>✔</td>
<td>x?</td>
<td>x?</td>
</tr>
<tr>
<td>Refinement</td>
<td>✔</td>
<td>x?</td>
<td>?</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Quantitative</td>
<td>✔</td>
<td>✔</td>
<td>✔?</td>
<td>x?</td>
<td>✔?</td>
</tr>
<tr>
<td>Qualitative</td>
<td>✔</td>
<td>◄?</td>
<td>✔</td>
<td></td>
<td>◄?</td>
</tr>
<tr>
<td>Time</td>
<td>✔</td>
<td>◄</td>
<td>✔</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

- ✔ Supported
- x Not supported
- ◄ Partially supported
- ? Not well-defined