Time in the UML profile for MARTE

Charles André
University of Nice-Sophia Antipolis / CNRS & INRIA
MARTE

- Modeling &
- Analysis of
- Real-Time and
- Embedded
- systems
An embedded system is an engineering artifact involving computation that is subject to physical constraints. 

Henzinger, Sifakis[1].
Overview

- Model-Driven Development
- SPT, UML 2 and Time
  - UML::CommonBehaviors::SimpleTime
- the MARTE Time domain view
  - a.k.a. the MARTE Time meta-model
  - Concepts and relationships
- the MARTE Time sub-profile
  - a.k.a. UML view
Model-Driven Development (in Software Engineering)

Hailpern & Tarr [2]

- The standard *laissez faire* approach to programming that many practitioners learned must be replaced by a more disciplined engineering methodology.

- **Model-driven development (MDD)** is a software engineering approach consisting of the application of *models* and *model technologies* to raise the level of abstraction at which developers create and evolve software, with the goal of both simplifying (making easier) and formalizing (standardizing, so that automation is possible) the various activities and tasks that comprise the software life cycle.

- The **Object Management Group, Inc. (OMG™)** defines a particular realization of MDD using the term **Model Driven Architecture® (MDA®)**.
  - Platform Independent Model (PIM)
  - Platform Specific Model (PSM)

- Other MDD approaches:
  - Agile Model-Driven Development,
  - Domain-Oriented Programming, and
  - Microsoft’s Software Factories.
Model-Driven Development (in Electronic Design)

A. Sangiovanni-Vincentelli [3] from electronic design to other systems (automotive, avionics, automation,...)

- **System-level design (SLD):** approach
- **Platform-based design (PBD):** methodology
  - Platform: a library of components
  - Design process: meet-in-the-middle
Why Model Driven Engineering is Needed?


- To deal with complexity of systems development
  - Abstract a problem to focus on some particular points of interest
    → improve understandability of a problem
  - Possible set of nearly independent views of a model
    - Separation of concerns (e.g. “Aspect Oriented Modeling”)
  - Iterative modeling may be expressed at different level of fidelity

- To minimize development risks
  - Through analysis and experimentation performed earlier in the design cycle
  - Enable to investigate and compare alternative solutions

- To improve communication ...
  ... to foster information sharing and reuse!
  - A model is often best suited than a long speech!

- To focus on specific domain expertise while developing software system
  - Domain Specific Language
Characteristics of Useful Models

- **Abstract**
  - Emphasize important aspects while removing irrelevant ones

- **Understandable**
  - Expressed in a form that is readily understood by observers

- **Accurate**
  - Faithfully represents the modeled system

- **Predictive**
  - Can be used to answer questions about the modeled system

- **Inexpensive**
  - Much cheaper to construct and study than the modeled system

from B. Selic’s presentation at the Summer School MDD for DRES 2004
(Brest, September 2004)
Our mission

- To define a **UML profile** for RT & E systems

**Goals:**
- to add capabilities to UML for model-driven development of Real Time and Embedded Systems (RTES);
- to provide support for specification, design, and verification/validation stages;
- to replace the UML profile for Schedulability, Performance and Time (SPT).

**Some requirements** (from the RFP for MARTE)
- to support independent modeling of both software or hardware parts of RT/E systems and the relationships between them.
- to provide modeling constructs covering the development process of RT/E systems.

**AOSTE main contributions to MARTE:**
- The **Time** chapter
- The **Allocation** chapter

The subject of this presentation
Pros & cons for UML profiles

B. Selic [5]
Approach to Domain-specific modeling language design

- **pros**
  - Reuse of language infrastructure (tools, specifications)
  - Require less language design skills
  - Allow for new (graphical) notation of extended stereotypes
  - A profile can define model viewpoints

- **cons**
  - Require a good knowledge of the UML meta-model
  - Constrained by UML meta-model
  - For OMG UML profiles: OMG standardization process
OMG Standardization Process

Final Adopted Specification Publication: August 6, 2007
Comments Due: December 22, 2007
Recommendations and Report Deadline: July 4, 2008

Final Adopted Specification Publication: August 6, 2007
Comments Due: December 22, 2007
Recommendations and Report Deadline: July 4, 2008

Charles André - University of Nice-Sophia Antipolis
Another look at Time

- Embedded system models very often consists of a predefined set of application functions, and of execution platforms on which to allocate these functions.
  - Application elements are increasingly componentized, may coexist and possibly cooperate concurrently.
  - Execution platforms increasingly comprise parallel resources for both communications and computations.

- The design challenge in embedded system modeling is then to provide model-level compilation techniques that provide support for both spatial distribution and temporal scheduling of applications onto platforms (collectively called allocation). This approach is therefore akin to system level design techniques.

- UML, hardly formalizes its real-time aspects.

- The primary objective of the Time sub-profile in MARTE was to provide basic and advanced time modeling concepts, with interpretation inside the UML modeling level, not outside. These time-related concepts could then be used to build various Models of Computation and Communication (MoCC).
Logical time in design

- Time as considered in design can be of physical or logical nature.
- Physical time is continuous, but can usually be discretized into chronometric time under appropriate assumptions.
- Logical time is less often recognized in itself as an explicit modeling concept.
  - e.g., Processing and execution steps performed at the rate of a processor cycle (which may vary according to power consumption management), or triggered by successive occurrences of an external event (such as completion of an engine revolution).
- Often the allocation process may be perceived as this:
  - asynchronous concurrent application components are each considered as being governed by their own (local) logical clock, connected to appropriate events;
  - the allocation itself consists in fitting these various clock threads onto a single (or at least more correlated) synchronous clock, subject to constraints of various sources abstracted from physical time properties and requirements.
- The transformation and analysis steps involved in the proper mapping are (at least implicitly) dealing with scheduling objects that are relations between logical and physical clocks attached to the various processing. MARTE Time profile is meant exactly to represent that.
Overview

- Model-Driven Development
- SPT, UML 2 and Time
  - UML::CommonBehaviors::SimpleTime
- the MARTE Time domain view
  - a.k.a. the MARTE Time meta-model
  - Concepts and relationships
- the MARTE Time sub-profile
  - a.k.a. UML view
UML profile for Schedulability, Performance, and Time (SPT)

- OMG UML profile formal/05-01-02 (v1.1)
- Based on UML 1.4
- Dealing with time and resources
- Quantitative time information
- Concepts
  - Instant, duration
  - Event bound to time, stimuli
- Timing mechanisms & services

To be aligned to UML 2
Metric time
UML::CommonBehaviors::SimpleTime

- UML2 adds **new metaclasses** to represent
  - Time
  - Duration
  - Observation (of time passing)
  - Some forms of time constraints
- **Simple** (even simplistic) model of time
- **Advice:** *Use a more sophisticated model of time provided by an appropriate profile*, if needed. [UML superstructure, chapter 13]

e.g., MARTE

Charles André - University of Nice-Sophia Antipolis
SimpleTime::TimeEvent

CommonBehaviors::
Communications::
Event

TimeEvent

isRelative: Boolean

0.1
when
1

Classes::
Kernel::
ValueSpecification

Absolute/relative specification

Time specification
TimeEvent - usage (1)

UML state machine = behavior

Specification of a time-trigger

Informal semantics

On

Off

after 10

stm blinker

after 10

Charles André - University of Nice-Sophia Antipolis
Metaclasses involved in the modeling of a transition triggered by a TimeEvent.

Simple annotation → complex implied structure.
SimpleTime::Observation

Classes::
Kernel::
NamedElement

DurationObservation

Observation

TimeObservation

firstEvent: Boolean

1 event

1..2 event

firstEvent: Boolean[0..2]
Observation - usage (1)

MOS stands for MessageOccurrenceSpecification
Instance model of the time constraint: receive CardOut in \{t .. t+d\}

Simple annotation → complex implied structure
Overview

- Model-Driven Development
- SPT, UML 2 and Time
  - UML::CommonBehaviors::SimpleTime
- the MARTE Time domain view
  - a.k.a. the MARTE Time meta-model
  - Concepts and relationships
- the MARTE Time sub-profile
  - a.k.a. UML view
Concepts in MARTE::Time (1)

- **Time structure** =
  - set of time bases + time structure relations
  - → Partially ordered set of instants

- **Access to time** = **Clock**

- **Principle:** associate Clocks with model elements
  - Behavioral elements → TimedEvent, TimedProcessing
  - Constraints → TimedConstraint
  - Data types and values → TimedValue
Concepts in MARTE::Time (2)

- **Concepts**
  - **Time bases**
  - **Multiple Time Bases**
  - **Instants**
  - **Time structure relations**

- **Concepts**
  - **Clocks**
  - **Logical clocks**
  - **Chronometric clocks**
  - **Current time**

- **Concepts**
  - **Timed elements**
  - **Timed events**
  - **Timed actions**
  - **Timed constraints**
Time Structure

MultipleTimeBase
= Ensemble de TimeBases + Hierarchy + Constraints

TimeBase = oset of instants

MultipleTimeBase

TimeBase

Instant

date: Real

Relationships over instants of different TBs

Relationships over TBs
Access to Time: Clock

Units associated to a clock

Event occurring at each clock ticking

Access to the time structure
Implicit reference to physical time

Possible reference to a repetitive event

NFPs measured against a reference clock
Time Values

A TimeValue has a unit (default = clock unit)

A TimeValue must reference a clock

Instant/Duration two distinct concepts

Charles André - University of Nice-Sophia Antipolis
Timed Entities: TimedElement

The unifying concept: a \textit{TimedElement} = a \textit{ModelElement} + a \textit{Clock}
Timed Entities: TimedEvent

- **CoreElements::**
  - Causality::
  - RunTimeContext::
  - EventOccurrence

- **SimultaneousOccurrenceSet**
  - occSet
  - 0..*

- **TimedEvent**

- **TimedEventOccurrence**
  - 0..1
  - 1..*
  - at

- **InstantValue**

- **Provision for simultaneity**

- **Facility to specify multiple occurrences**

- **TimedElement**

- **Event**
  - isRelative: Boolean
  - repetition: Integer [0..1]

- **TimeValueSpecification**
  - 1
  - when

- **DurationValueSpecification**
  - 0..1
  - every
Timed Entities: Timed Processing

```
CoreElements::
Causality::
CommonBehavior::
Behavior

TimedBehavior

CoreElements::
Causality::
Communication::
Request

TimedMessage

CoreElements::
Causality::
CommonBehavior::
Action

TimedAction

TimedProcessing

DurationValueSpecification

Event

TimedElement

Delay
```

0.1 duration

start 0.1 finish 0.1
Underlying mathematical model (1)

Clock := \langle \mathcal{I}, \prec, \mathcal{D}, \lambda, u \rangle

- \mathcal{I} is a set of instants,
- \prec is a (total) pre-order on \mathcal{I}; \langle \mathcal{I}, \prec \rangle is an oset
- \mathcal{D} is a set of labels
- \lambda : \mathcal{I} \rightarrow \mathcal{D} is a labeling function
- u is a symbol (unit)
Underlying mathematical model (2)

TimeStructure::= \langle C, R, D, \lambda \rangle \quad \text{where}

- $C$ is a set of clocks,
- $R$ is a relation on $\bigcup_{a, b \in C, a \neq b} (I_a \times I_b)$,
- $D$ is a set of labels,
- $\lambda : I_C \to D$ is a labeling function.

$(R_t)_{t \in \{co, \text{pred}, \text{spred}\}}$ is a partition of $R$:

1) $R_{co}, R_{\text{pred}}, R_{\text{spred}}$ are pairwise disjoint relations:

$(\forall (i, j) \in R, \forall t \in \tau)(i, j) \in R_t \Rightarrow (\forall t' \in \tau \setminus \{t\})(i, j) \notin R_{t'}$

2) $R = R_{co} \cup R_{\text{pred}} \cup R_{\text{spred}}$.

$D$ is usually a Cartesian product of domains $D = \bigotimes_{k} D_k$.
Underlying mathematical model (3)

Let $\mathcal{I} = \bigcup_{a \in C} \mathcal{I}_a$ be the set of all the instants of the clocks in the time structure.

Let $\equiv \subset \mathcal{I} \times \mathcal{I}$ be the reflexive transitive closure of $\mathcal{R}_{co}$.

Let $\mathcal{I}_c = \mathcal{I} / \equiv$ be the quotient set of $\mathcal{I}$ by $\equiv$.

$[j] \in \mathcal{I}_c$ denotes the equivalence class of $j \in \mathcal{I}$.

Let $\preceq \subset \mathcal{I}_c \times \mathcal{I}_c$ be the smallest relation on $\mathcal{I}_c \times \mathcal{I}_c$ such that for all $i, j \in \mathcal{I}$:

1) $(\exists a \in C)(i, j \in \mathcal{I}_a) \wedge (i \prec_a j) \Rightarrow [i] \preceq [j],$

2) $(\exists a, b \in C)(i \in \mathcal{I}_a) \wedge (j \in \mathcal{I}_b) \wedge (i \mathcal{R}_{\text{pred}} j) \Rightarrow [i] \preceq [j],$

3) $(\exists a, b \in C)(i \in \mathcal{I}_a) \wedge (j \in \mathcal{I}_b) \wedge (i \mathcal{R}_{\text{spred}} j) \Rightarrow [i] \times [j]$.

where $\preceq = \equiv \setminus \text{Id}$

The time structure $T = (C, \mathcal{R}, \mathcal{D}, \lambda)$ is well-structured iff $\langle \mathcal{I}_c, \preceq \rangle$ is a partially ordered set (i.e.; $\mathcal{R}$ does not cause cycles).

Charles André - University of Nice-Sophia Antipolis
Overview

- Model-Driven Development
- SPT, UML 2 and Time
  - UML::CommonBehaviors::SimpleTime
- the MARTE Time domain view
  - a.k.a. the MARTE Time meta-model
  - Concepts and relationships
- the MARTE Time sub-profile
  - a.k.a. UML view
Providing extensions to UML

- Through a UML profile
  - New Stereotypes
- Facilities
  - Model libraries
  - Dedicated languages (especially for expressions)
Two other subprofiles of MARTE

dependencies

Two other subprofiles of MARTE

User’s model library
Central stereotypes: ClockType & Clock

**Chronometric clock** → "physical" time; units={s, ms, us, …}

**Logical clock** → any repetitive event; units={tick} U physicalUnits

- Accepted units
- Default unit

<table>
<thead>
<tr>
<th>stereotypes properties: Special semantics</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>isLogical</th>
<th>nature</th>
<th>discrete</th>
<th>dense</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>Logical clock</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>Chronometric clock</td>
<td>discrete</td>
<td>dense</td>
</tr>
</tbody>
</table>
Clock and TimedElement

- « metaclass » UML::Classes::Kernel::Package
- « stereotype » TimedDomain
- « stereotype » NFPs::Unit
- « metaclass » UML::Classes::Kernel::InstanceSpecification
- « stereotype » Clock
- « stereotype » TimedElement
- « metaclass » UML::Classes::Kernel::Class
- « stereotype » ClockType

nature: TimeNatureKind [1]
unitType: Enumeration [0..1]
isLogical: Boolean [1] = false
resolAttr: Property [0..1]
maxValAttr: Property [0..1]
offsetAttr: Property [0..1]
getTime: Operation [0..1]
setTime: Operation [0..1]
indexToValue: Operation [0..1]
TimedValueSpecification

- either Instant
- or Duration
Extending the TimeEvent metaclass of SimpleTime
Timed Processing

- « metaclass »
  - UML::Actions::Action
- « metaclass »
  - UML::CommonBehaviors::Behavior
- « metaclass »
  - UML::CommonBehaviors::Communication::Event
- « stereotype »
  - TimedProcessing
- « metaclass »
  - UML::Interactions::BasicInteractions::Message
- « stereotype »
  - TimedProcessing
  - UML::Classes::Kernel::ValueSpecification
- « stereotype »
  - TimedElement

Charles André - University of Nice-Sophia Antipolis
TimedObservation

« stereotype »
TimedElement

« stereotype »
TimedInstantObservation
obsKind:EventKind [0..1]

« stereotype »
TimedDurationObservation
obsKind:EventKind [0..2]

« metaclass »
UML::CommonBehaviors::SimpleTime::TimeObservation

« metaclass »
UML::CommonBehaviors::SimpleTime::DurationObservation
TimedConstraint & ClockConstraint

- « stereotype »
  - TimedConstraint
    - Interpretation: TimeInterpretationKind
- « stereotype »
  - NFPs::
    - NfpConstraint
- « stereotype »
  - ClockConstraint
- « stereotype »
  - TimedElement

Charles André - University of Nice-Sophia Antipolis
Time-related GRM stereotypes

Stereotypes defined in the Generic Resource Modeling sub-profile

- « stereotype » Resource
- « stereotype » Time::ClockType
- « stereotype » TimingResource
- « stereotype » TimerResource
- « stereotype » ClockResource

Resources for time management
Time-related libraries: TimeTypesLibrary

- « modelLibrary »
  TimeTypesLibrary

- « enumeration »
  TimeNatureKind
  discrete
dense

- « enumeration »
  TimeStandardKind
  TAI
UTC
Local
...
GPS

- « enumeration »
  TimeInterpretationKind
duration
instant

- « enumeration »
  EventKind
  start
finish
send
receive
consume
Time-related libraries: TimeLibrary

Two usual sets of Time Units

Model of ideal “physical time”

Templated DataType
Time-related NFP types

- **NFP_Duration**
  - `value`: Real
  - `unit`: TimeUnitKind
  - `precision`: Real

- **NFP_Real**
  - `value`: Real

- **NFP_DateTime**
  - `value`: DateTime

- **NFP_CommonType**
  - `expr`: VSL_Expression
  - `source`: SourceKind
  - `statQ`: StatisticalQualifierKind
  - `dir`: DirectionKind

- **NFP_Frequency**
  - `unit`: FrequencyUnitKind
  - `precision`: Real
Time Values: Concrete syntax

- **Simple time values**
  
  \[(\text{value}=3.5, \text{unit}=\text{ms}, \text{onClock}=\text{idealClk});\]
  
  3.5 ms on \text{idealClk};

- **Homogeneous expressions**
  
  \[(\text{value}=1.5, \text{unit}=\text{ms}, \text{onClock}=\text{idealClk}) +
  \text{(value}=150, \text{unit}=\text{us}, \text{onClock}=\text{idealClk});\]
  
  \[\rightarrow (\text{value}=1650, \text{unit}=\text{us}, \text{onClock}=\text{idealClk})\]

- **Heterogeneous expressions**
  
  \[
  \min (15 \text{ tick on } \text{prClk}, 5 \text{ ms on } \text{idealClk});
  \]

- **Additional capabilities with VSL**
  
  - Occurrence number, jitter,...
  
  - but implicitly on \text{idealClk}

**Tuple**, \emph{a la} VSL

**Short form**

Can be evaluated, because convFactor between units

Clock relation between \text{prClk} and \text{idealClk} must be provided
Time specific languages: VSL Time Constraints

$t_0$: observation of the message: start

$t_0$ is periodic, period 100ms with a jitter less than 5ms

- $s_d$: Data Acquisition
- $\text{Controller}$
- $\text{Sensor}$

Supported languages:

- VSL

Time constraints:

1. **VSL Time Constraints**
2. **Data Acquisition**: Controller
3. **Controller**
4. **Sensor**
5. **start()**: $\{\text{jitter}(t_0) < (5, \text{ms})\}$
6. **acquire()**: $\{d_1 < (1, \text{ms})\}$
7. **sendData(data)**: $\{(0, \text{ms})..(10, \text{ms})\}$
8. **@t2**
9. **@t3**
10. **constr1**: \( (t_{0[i+1]}-t_{0[i]}) > (100, \text{ms}) \)
11. **constr2**: \( t_3 < t_2 + (30, \text{ms}) \)

The figure illustrates the interactions and time constraints between the controller and sensor, highlighting the periodicity and time intervals for different operations.
Clock Constraint Specification

Each relation has a mathematical specification

Supported by the Clock Constraint Specification Language (CCSL)
Conclusion

- Enhanced time modeling within UML
  - Multi-clock time structure
  - Mathematical background
  - Effective profile (XMI provided)
- Model with mathematical background
- Still to be done: automatic clock relation usage in compilation/synthesis
Acknowledgments

- **Supports:**
  - *Carroll* initiative: Protes, Cortess
  - Project Usine Logicielle (pôle de compétitivité System@tic, Paris-Région) sub-project *OpenDevFactory*
  - RNTL Plate-forme *Open-eMbeDD*
References


