Time in the UML profile for MARTE

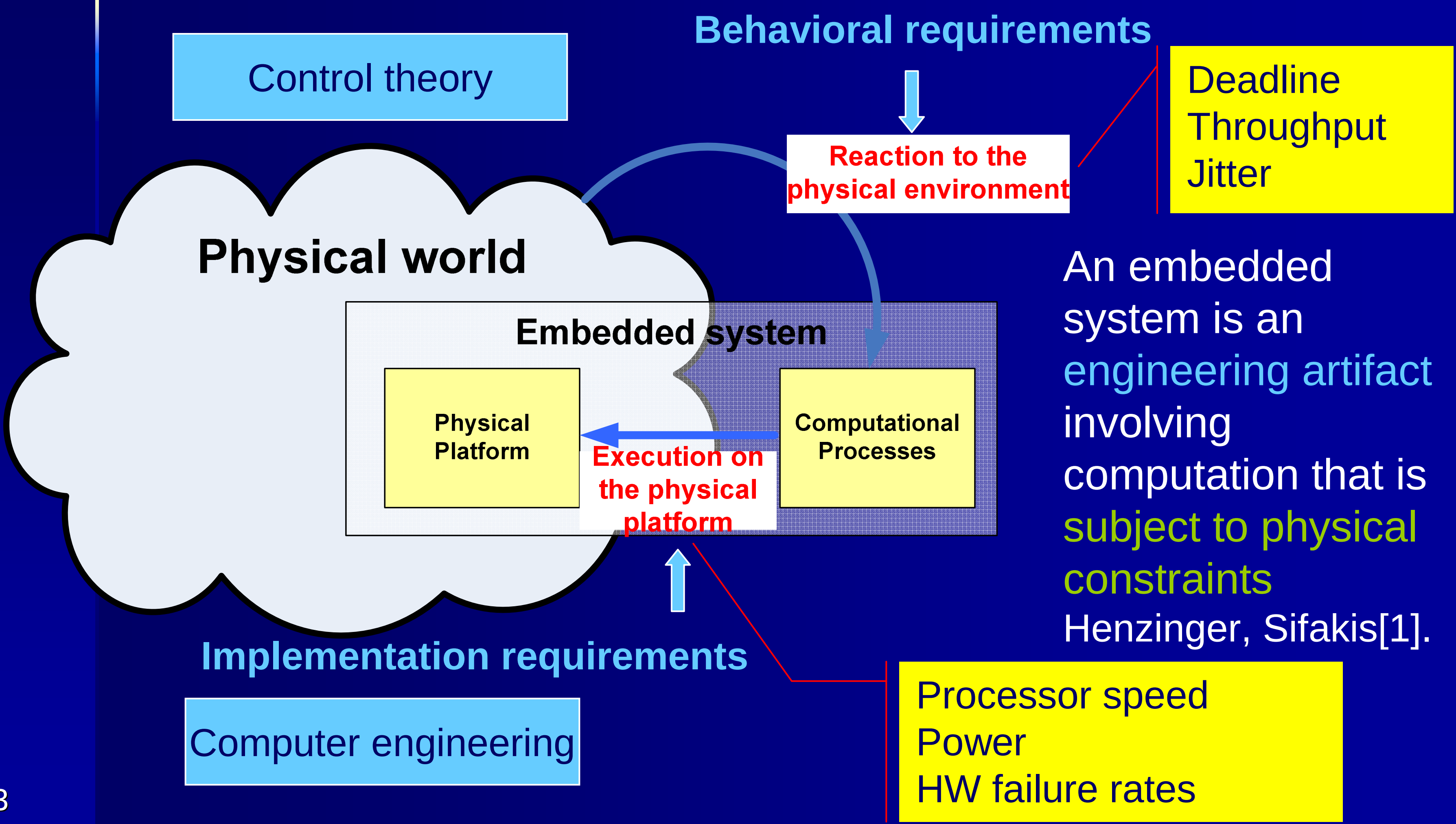
Charles André

University of Nice-Sophia Antipolis / CNRS & INRIA

MARTE

- Modeling &
- Analysis of
- Real-Time and
- Embedded
- systems

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)

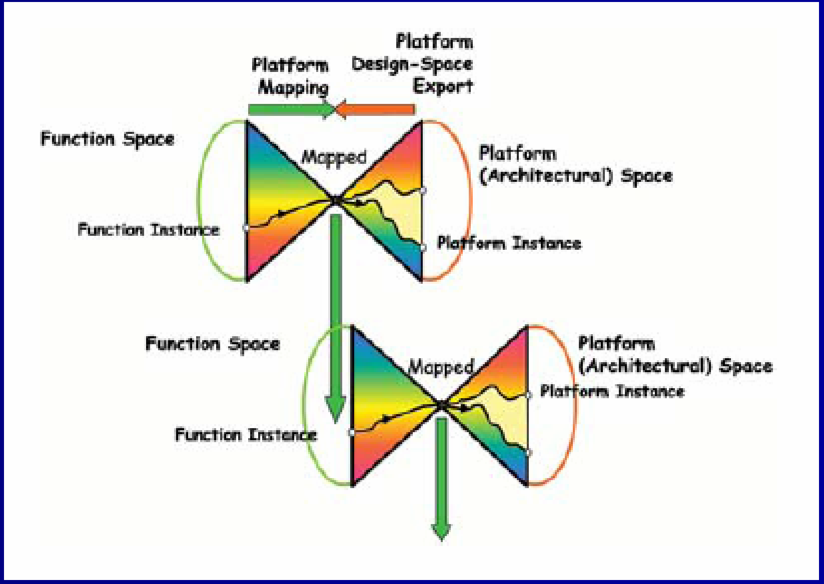
Hailpern & Tarr [2]

- 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?

S. Gérard, J. Medina, D. Petriu [4]

- 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 subject of this presentation
  - The Allocation chapter

Pros & cons for UML profiles

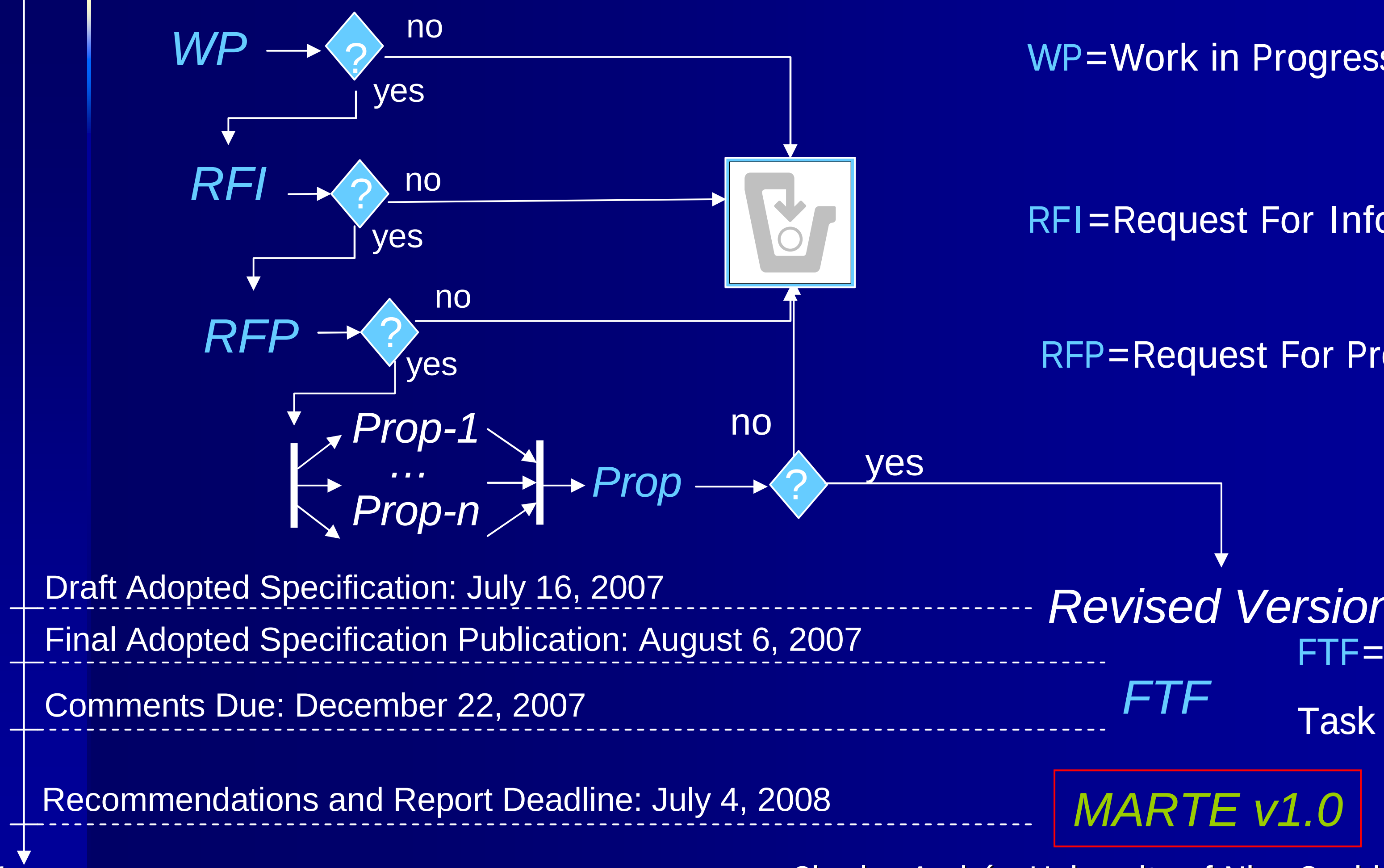
B. Selic [5]

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



WP=Work in Progress

RFI=Request For Information

RFP=Request For Proposal

FTF=Finalization Task Force

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)

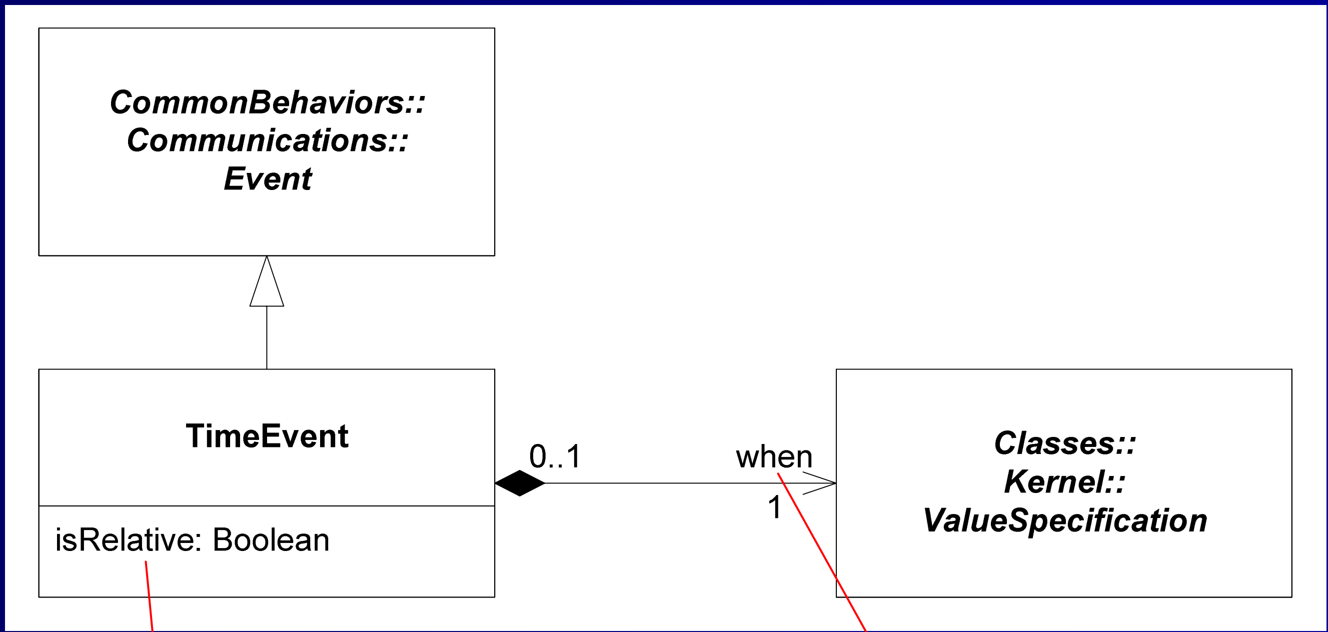
- OMG UML profile formal/05-01-02 (v1.1)
- Based on UML 1.4 — To be aligned to UML 2
- Dealing with time and resources
- Quantitative time information — Metric time
- Concepts
  - Instant, duration
  - Event bound to time, stimuli
- Timing mechanisms & services

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

SimpleTime::TimeEvent

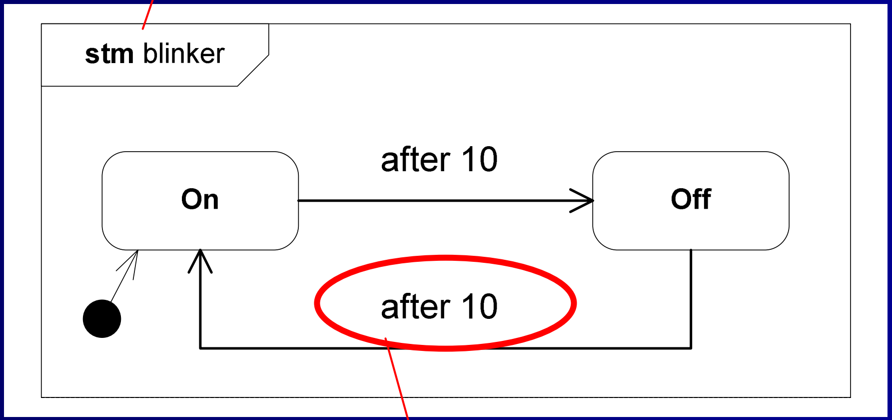


Time specification

Absolute/relative specification

TimeEvent – usage (1)

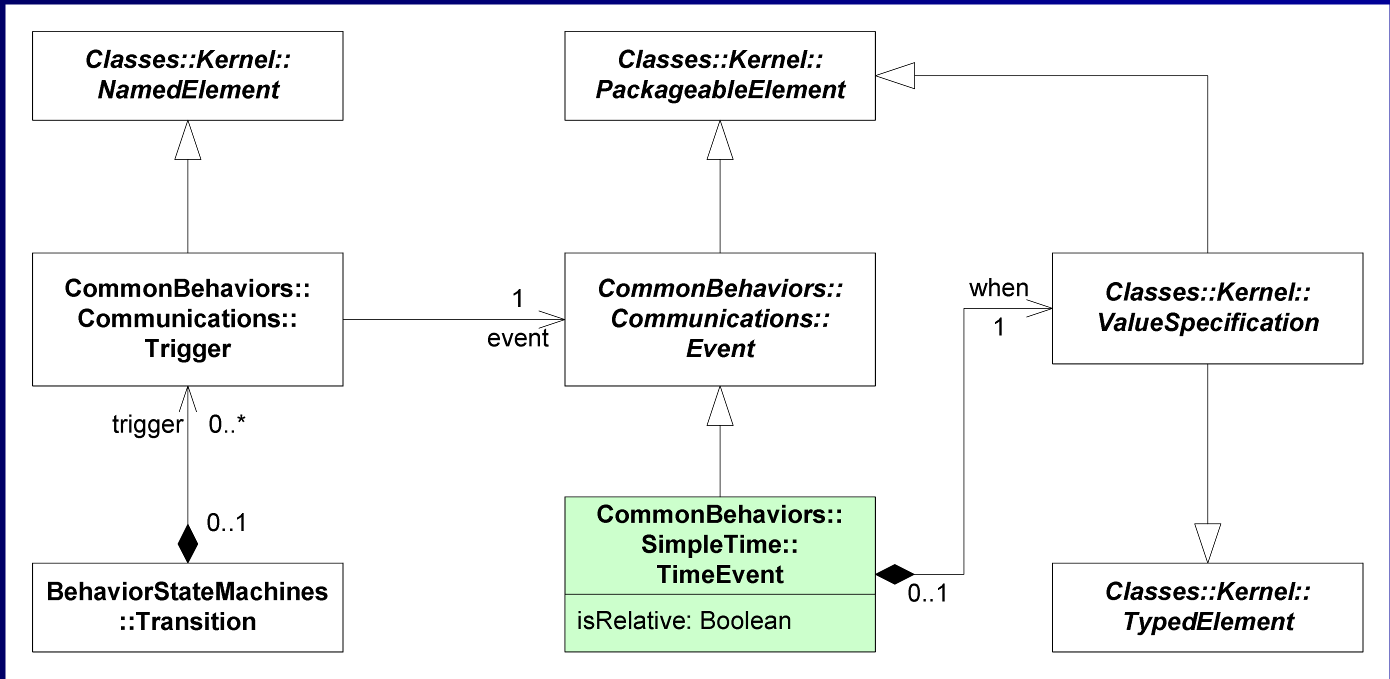
UML state machine = behavior



Specification of a time-trigger

Informal semantics

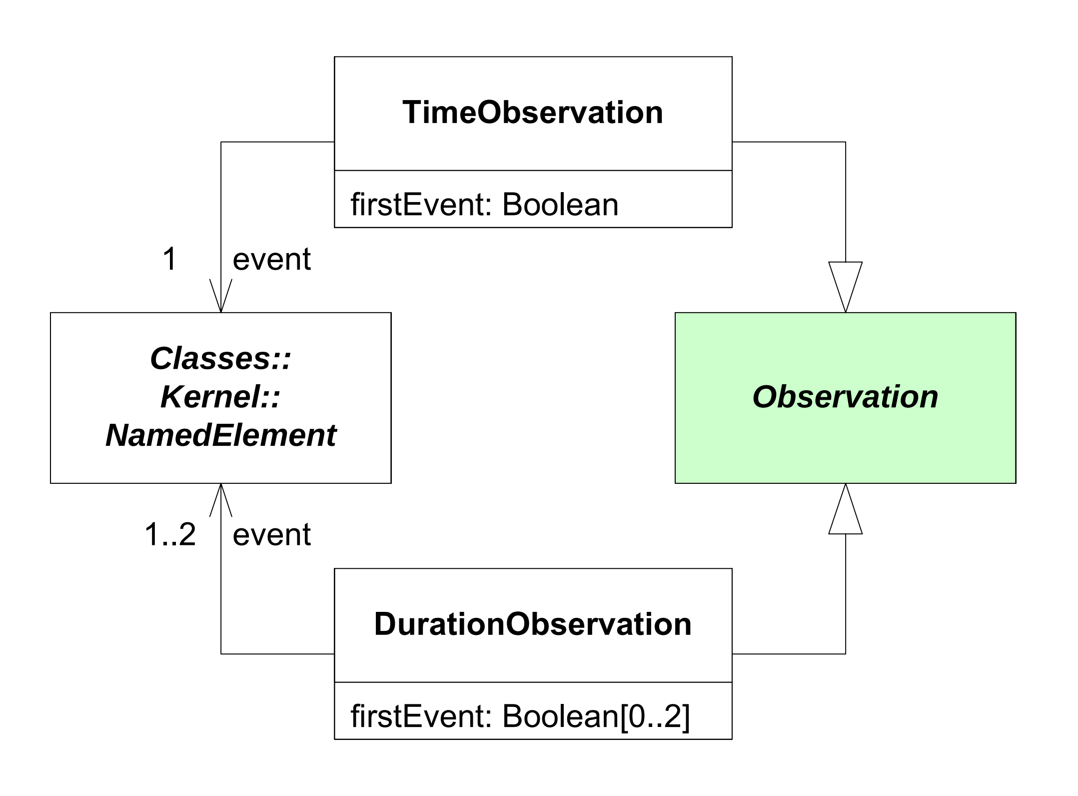
TimeEvent – usage (2)



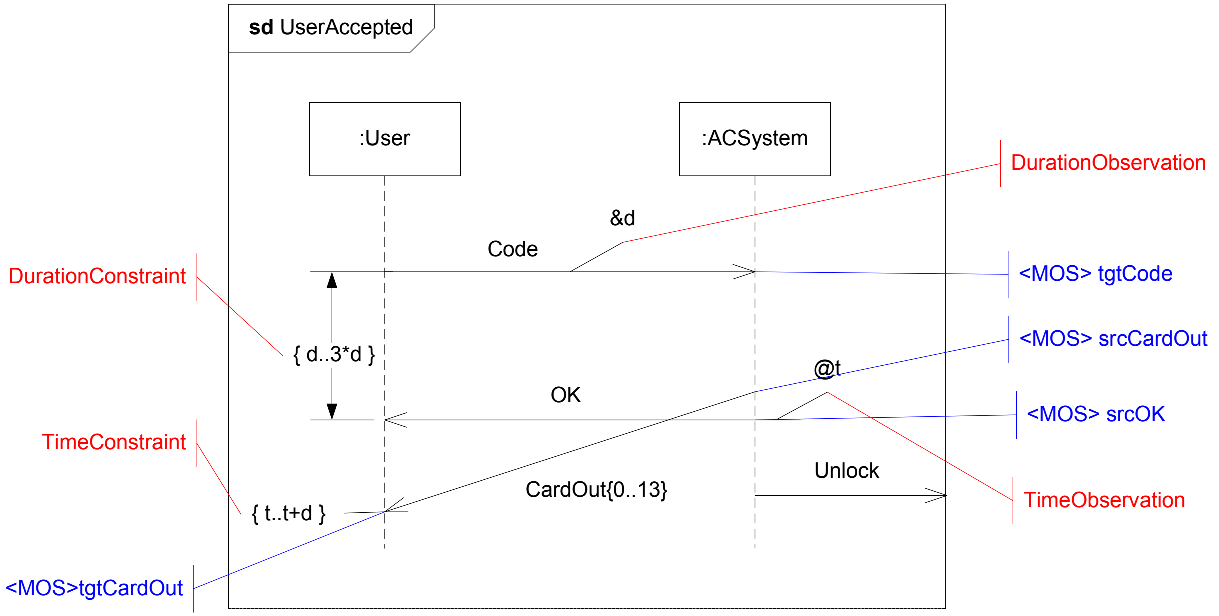
Metaclasses involved in the modeling of a transition triggered by a TimeEvent

Simple annotation → complex implied structure

SimpleTime::Observation

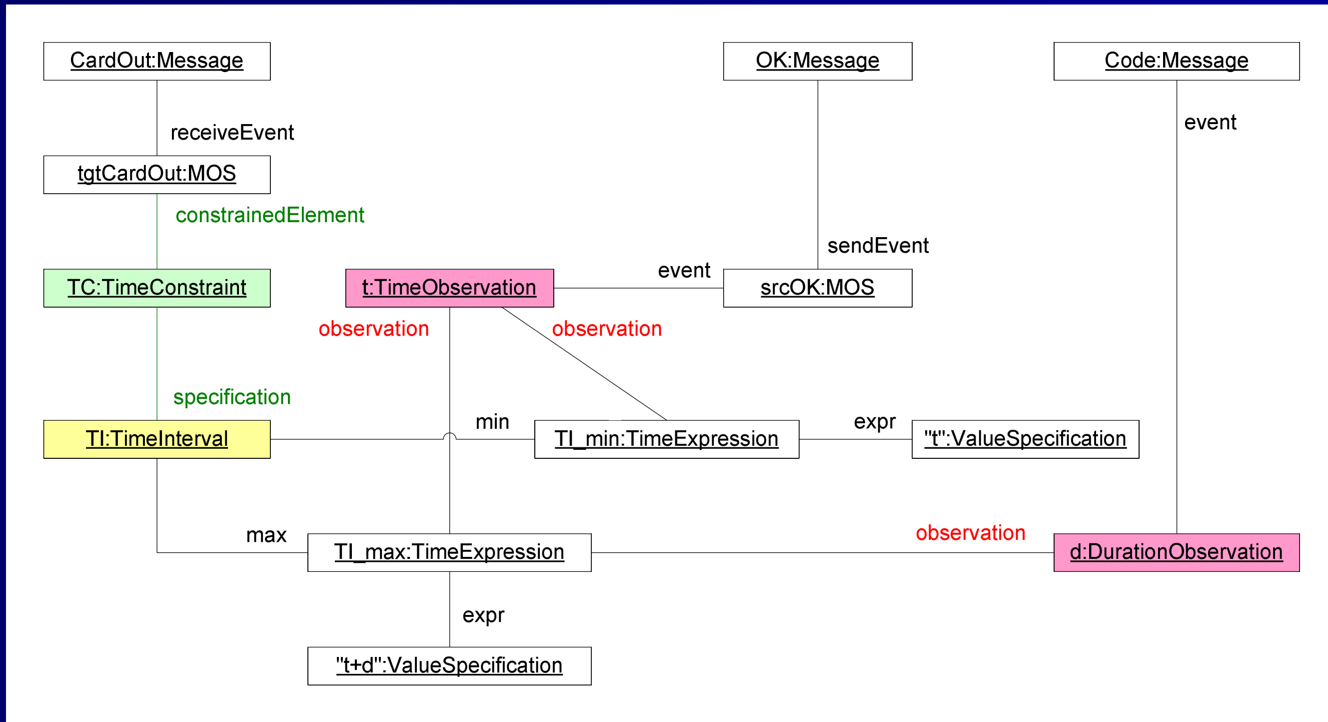


Observation – usage (1)



MOS stands for MessageOccurrenceSpecification

Observation – usage (2)



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

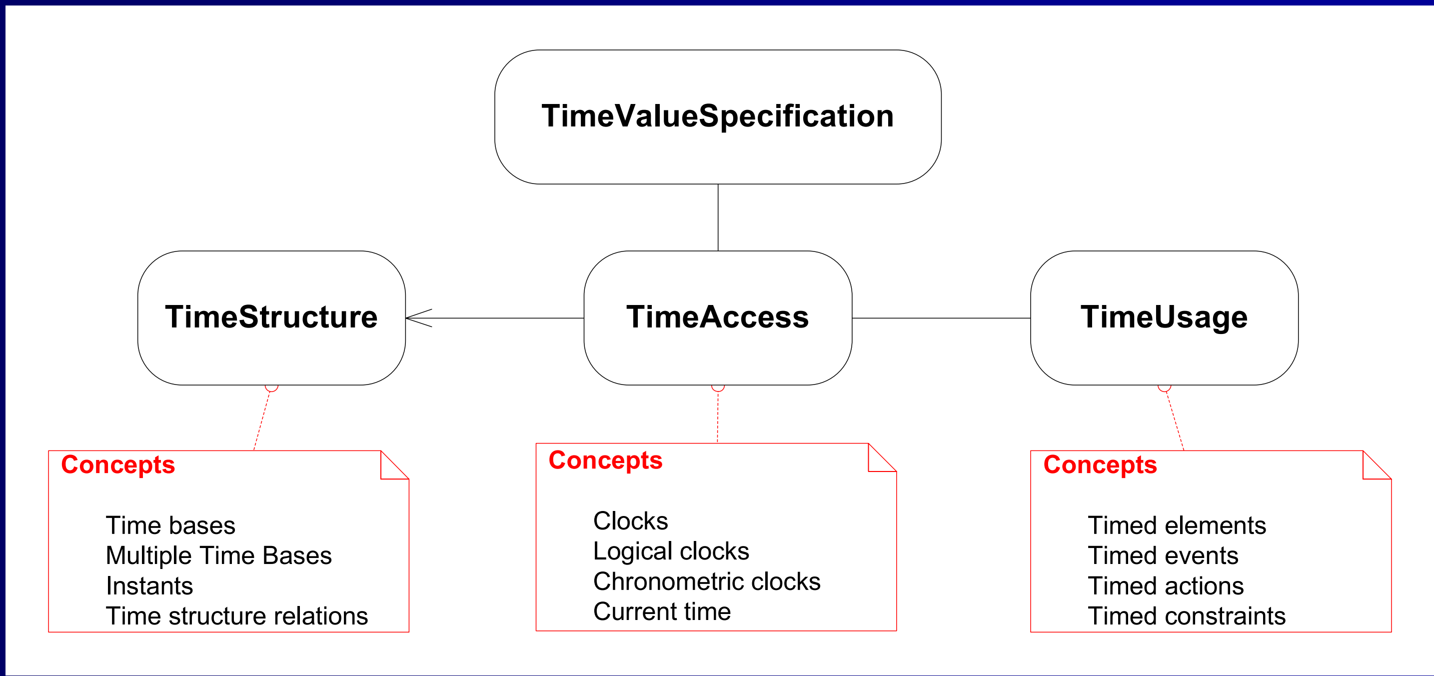
Concepts in MARTE::Time (1)

- Time structure =

  set of time bases + time structure relations

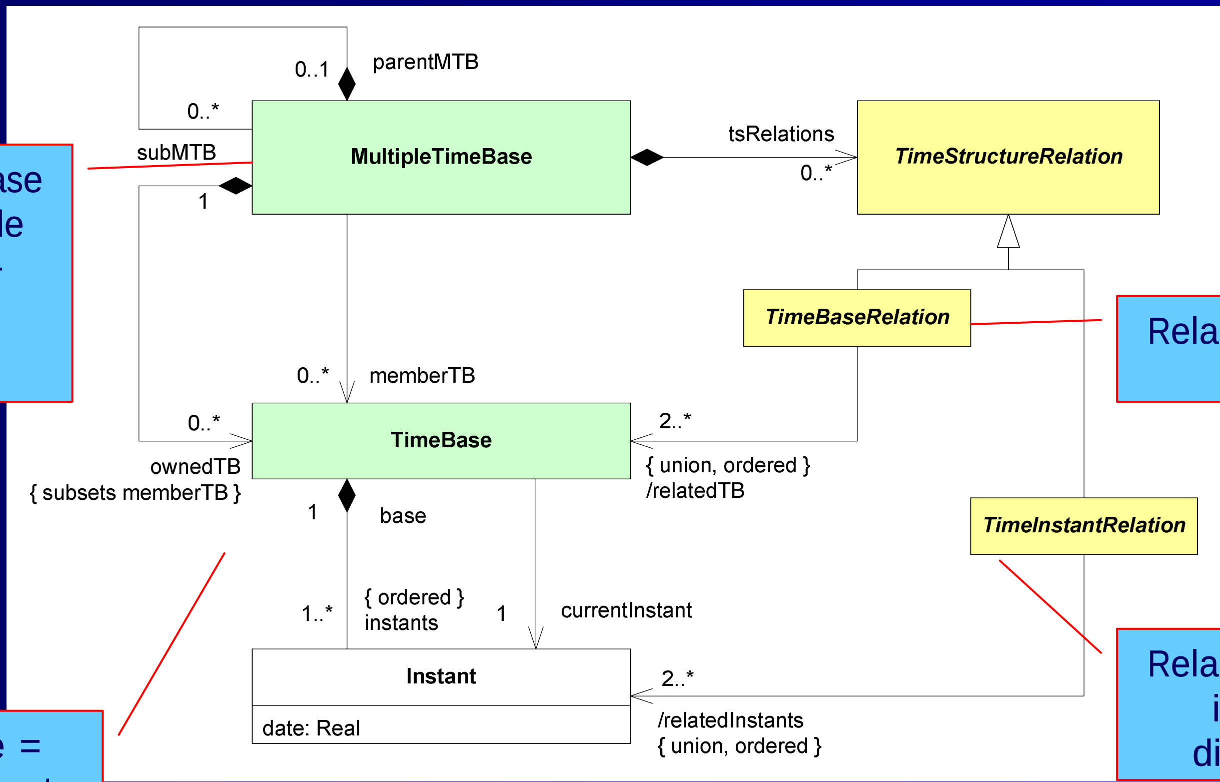
  → Partially ordered set of instants
- Access to time = Clock
- Principle: associate Clocks with model elements
  - Behavioral elements → TimedEvent, TimedProcessing
  - Constraints → TimedConstraint
  - Data types and values → TimedValue

Concepts in MARTE::Time (2)



Time Structure

MultipleTimeBase = Ensemble de TimeBases + Hierarchy + Constraints

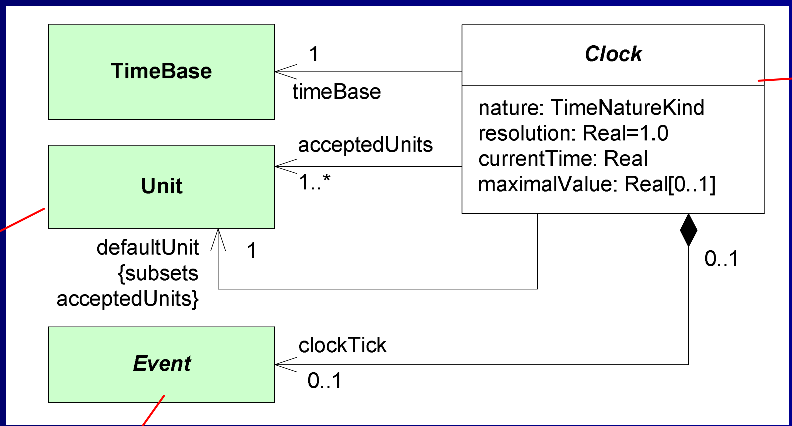


Relationships over TBs

Relationships over instants of different TBs

TimeBase = oset of instants

Access to Time: Clock



Access to the time structure

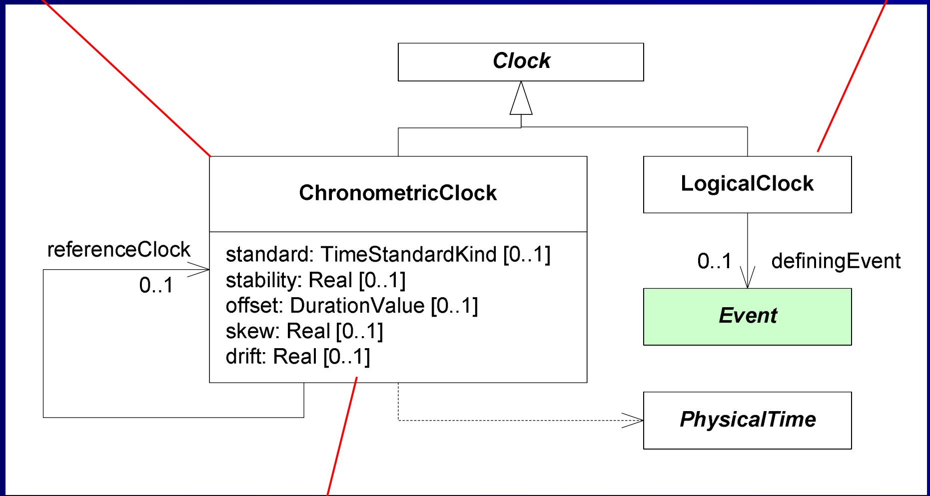
Units associated to a clock

Event occurring at each clock ticking

Chronometric/Logical Clocks

Implicit reference to physical time

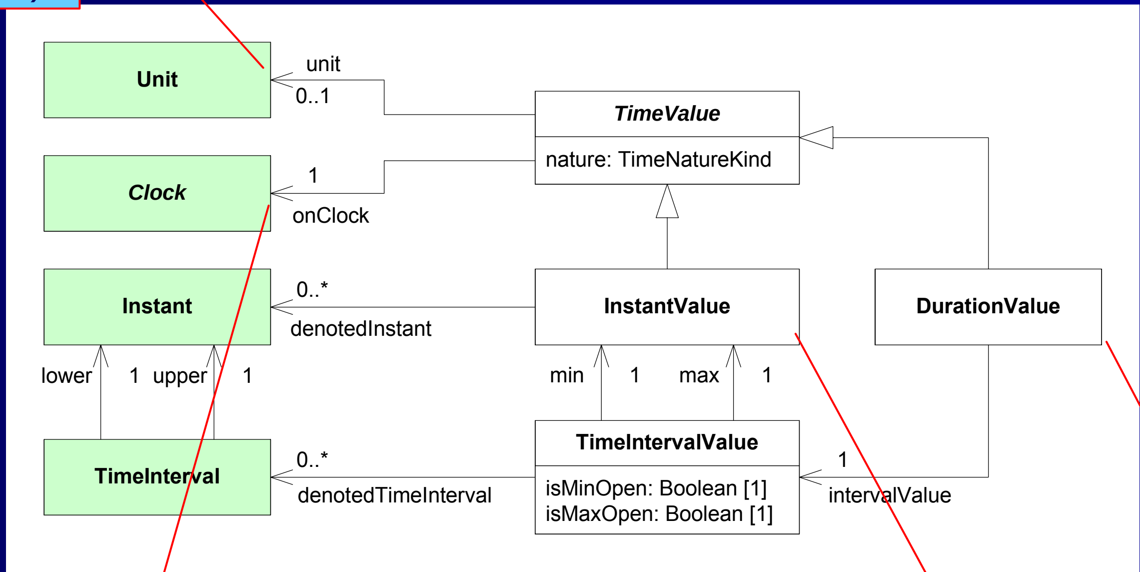
Possible reference to a repetitive event



NFPs measured against a reference clock

Time Values

A TimeValue has a unit (default= clock unit)

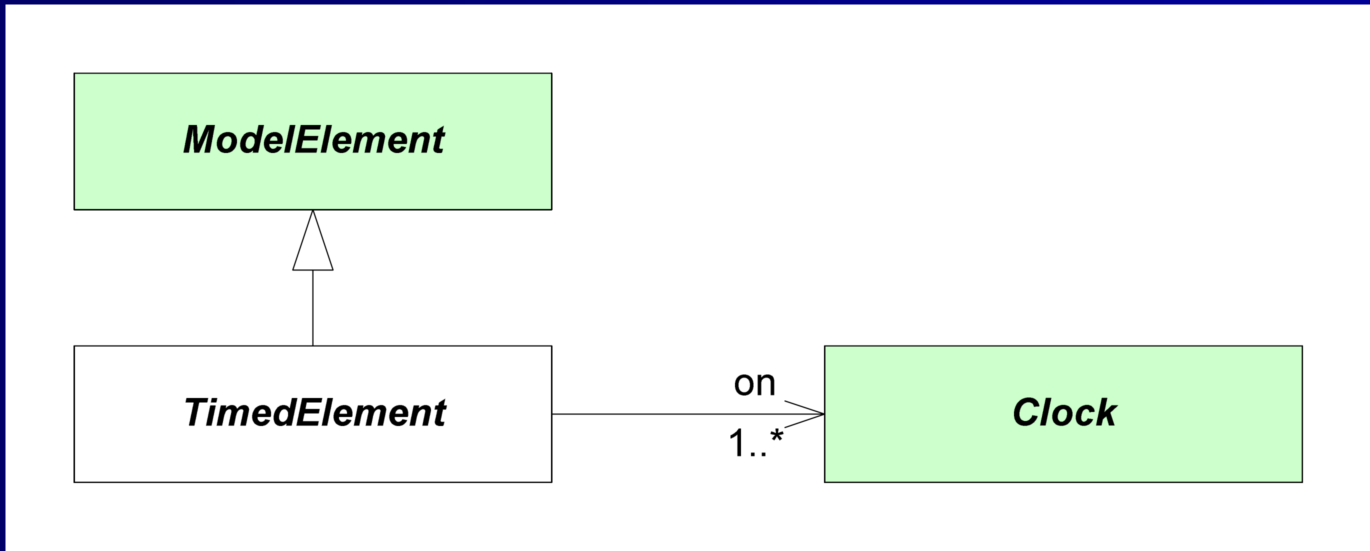


A TimeValue must reference a clock

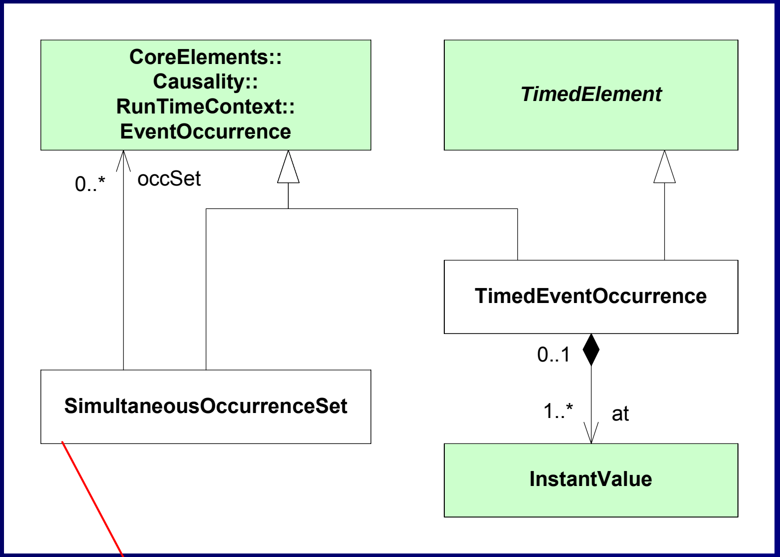
Instant/Duration two distinct concepts

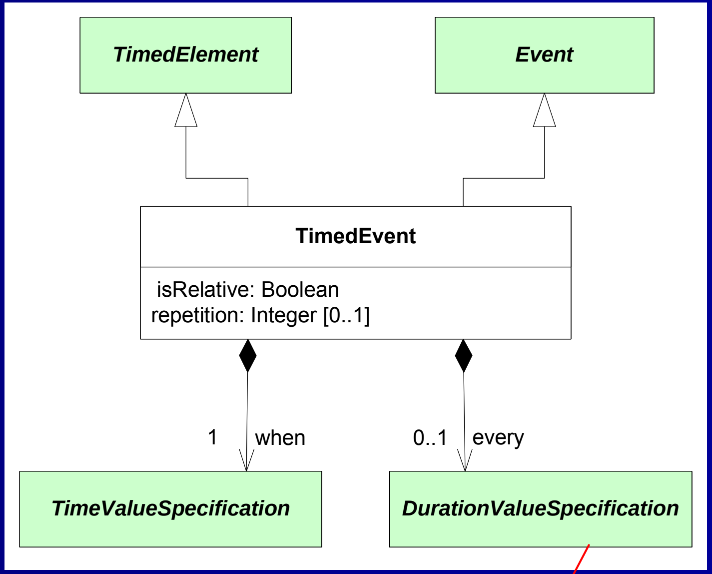
Timed Entities: TimedElement

The unifying concept: a TimedElement = a ModelElement + a Clock



Timed Entities: TimedEvent

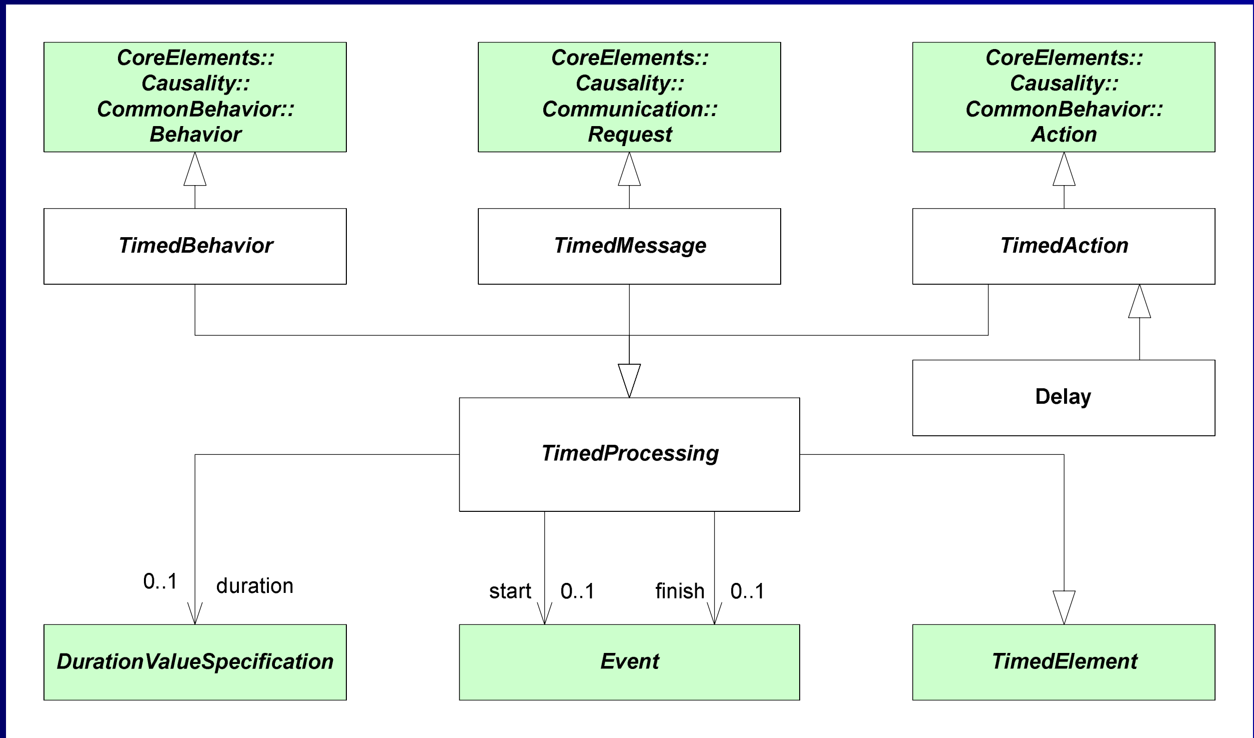




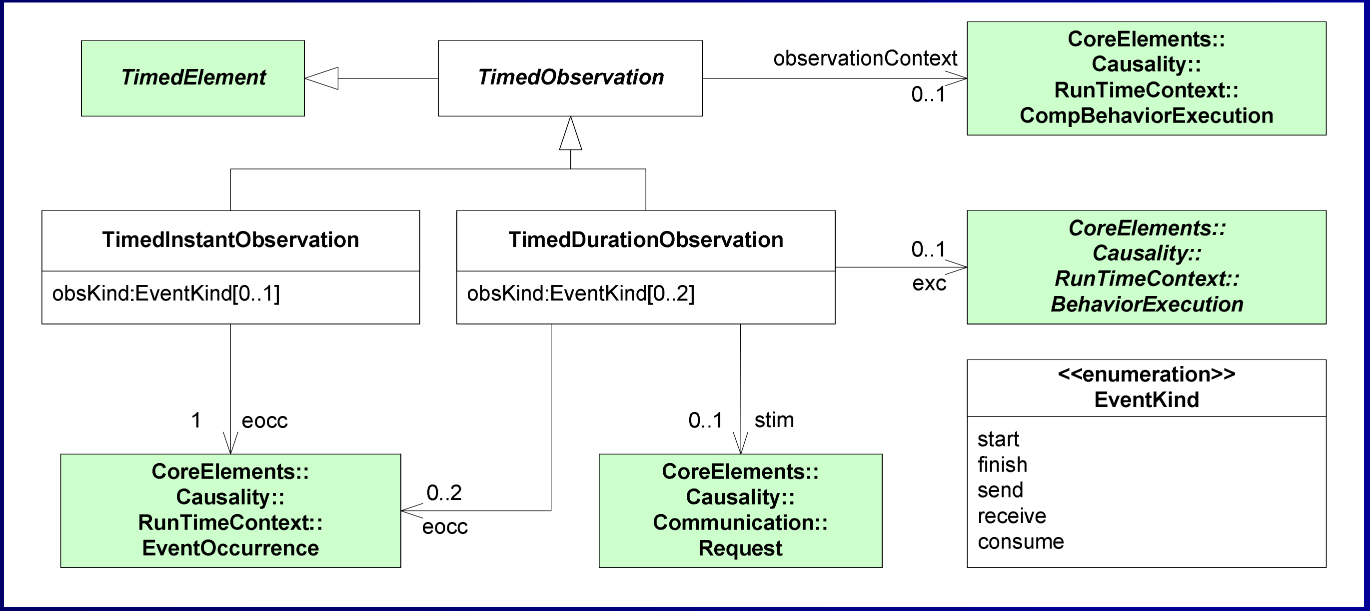
Provision for simultaneity

Facility to specify multiple occurrences

Timed Entities: TimedProcessing



Timed Entities: TimedObservation



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::= $\langle \mathcal{C}, \mathcal{R}, \mathcal{D}, \lambda \rangle$ where

$\mathcal{C}$ is a set of clocks,

$\mathcal{R}$ is a relation on $\bigcup_{a,b \in \mathcal{C}, a \neq b} (\mathcal{I}_a \times \mathcal{I}_b)$

$\mathcal{D}$ is a set of labels,

$\lambda : \mathcal{I}_{\mathcal{C}} \rightarrow \mathcal{D}$ is a labeling function.

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

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

$$(\forall (i,j) \in \mathcal{R}, \forall t \in \tau)(i,j) \in \mathcal{R}_t \Rightarrow (\forall t' \in \tau \setminus \{t\})(i,j) \notin \mathcal{R}_{t'}$$

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

$\mathcal{D}$ is usually a Cartesian product of domains $\mathcal{D} = \bigtimes_k \mathcal{D}_k$

Underlying mathematical model (3)

Let $\mathcal{I} = \bigcup_{a \in \mathcal{C}} \mathcal{I}_a$ 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 $\mathcal{R}_{\text{co}}$.

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

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

Let $\preccurlyeq \subset \mathcal{I}_{\mathcal{C}} \times \mathcal{I}_{\mathcal{C}}$ be the smallest relation on $\mathcal{I}_{\mathcal{C}} \times \mathcal{I}_{\mathcal{C}}$ such that

for all $i, j \in \mathcal{I}$:

1) $(\exists a \in \mathcal{C})(i, j \in \mathcal{I}_a) \wedge (i \prec_a j) \Rightarrow [i] \prec [j]$,
2) $(\exists a, b \in \mathcal{C})(i \in \mathcal{I}_a) \wedge (j \in \mathcal{I}_b) \wedge (i \mathcal{R}_{\text{pred}} j) \Rightarrow [i] \preccurlyeq [j]$,
3) $(\exists a, b \in \mathcal{C})(i \in \mathcal{I}_a) \wedge (j \in \mathcal{I}_b) \wedge (i \mathcal{R}_{\text{spred}} j) \Rightarrow [i] \prec [j]$

where $\prec = \preccurlyeq \setminus \text{Id}$

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

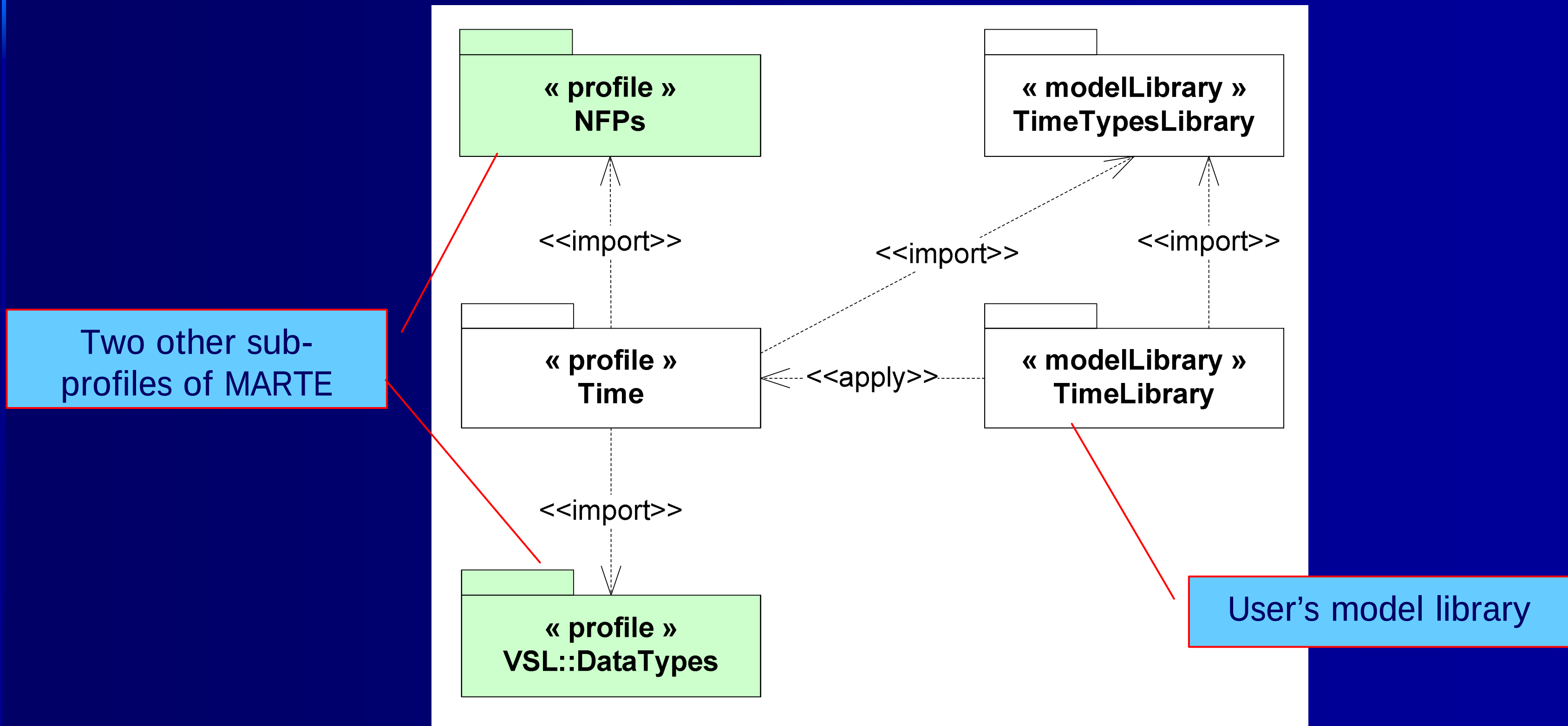
Overview

- 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)

Dependencies



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**

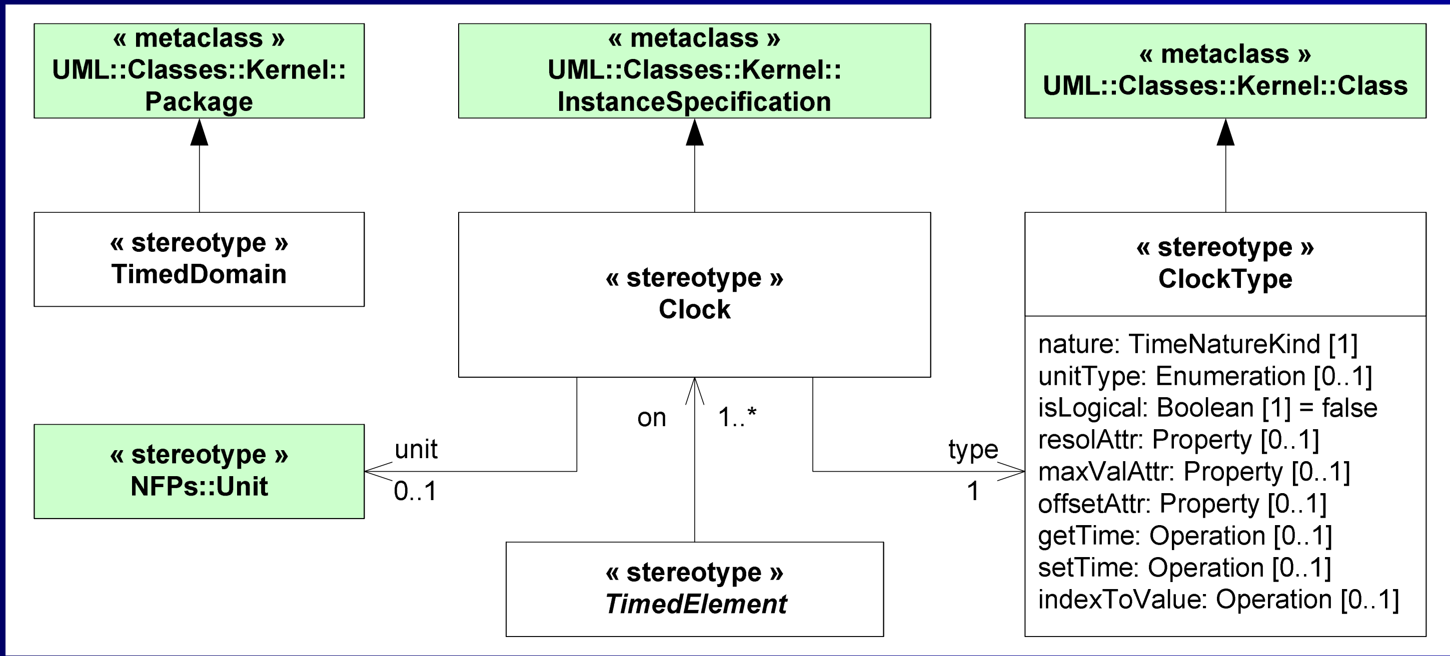
Stereotype properties:
Special semantics

+ optional

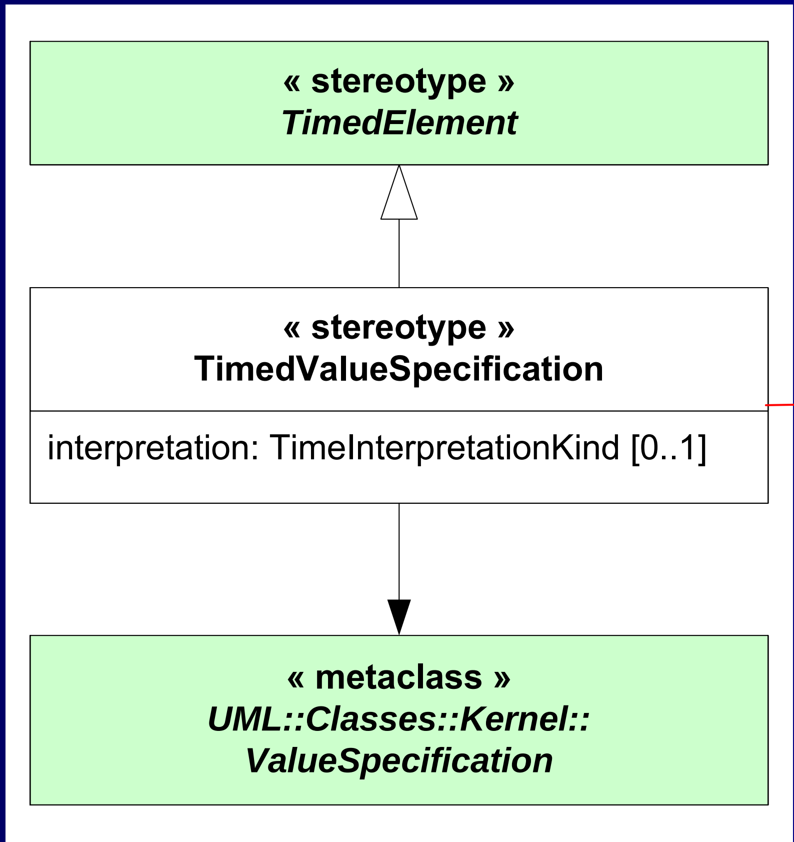
- **set of properties**
- **set of operations**

isLogical \ nature	discrete	dense
true	**Logical clock**	Not used
false	**Chronometric clock** discrete	**Chronometric clock** dense

Clock and TimedElement

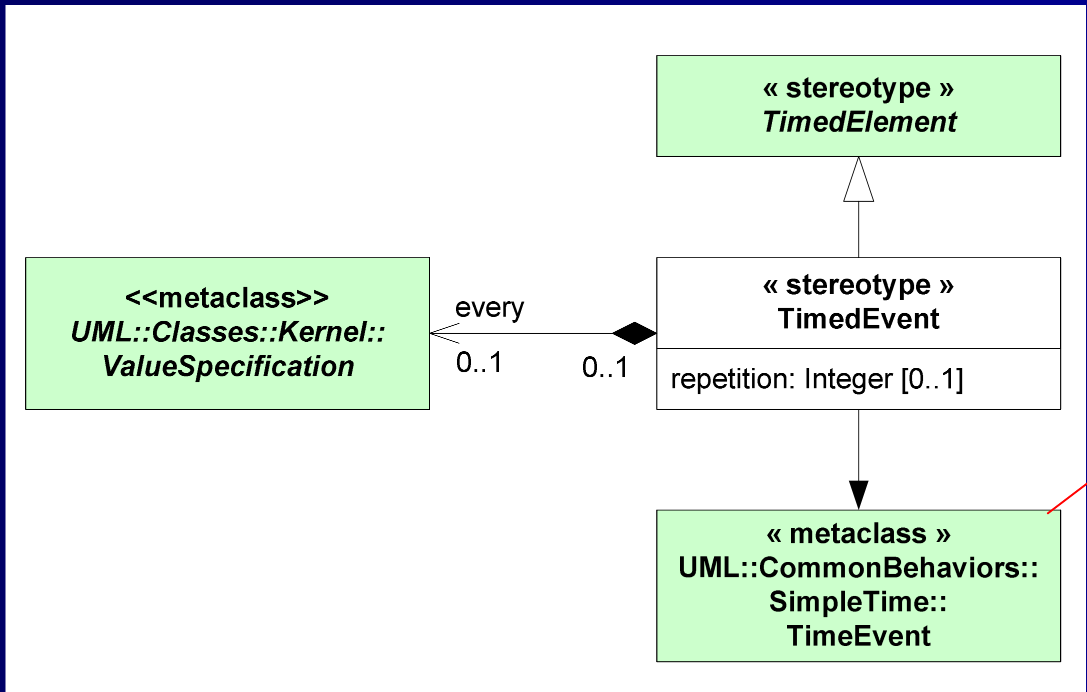


TimedValueSpecification



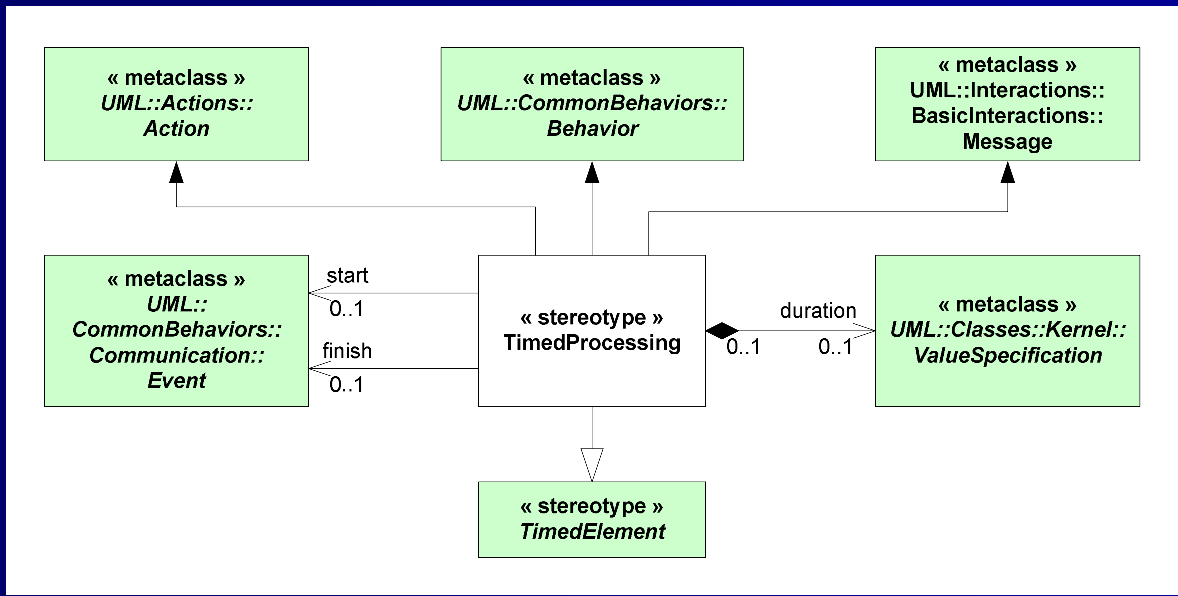
- either Instant
- or Duration

TimedEvent

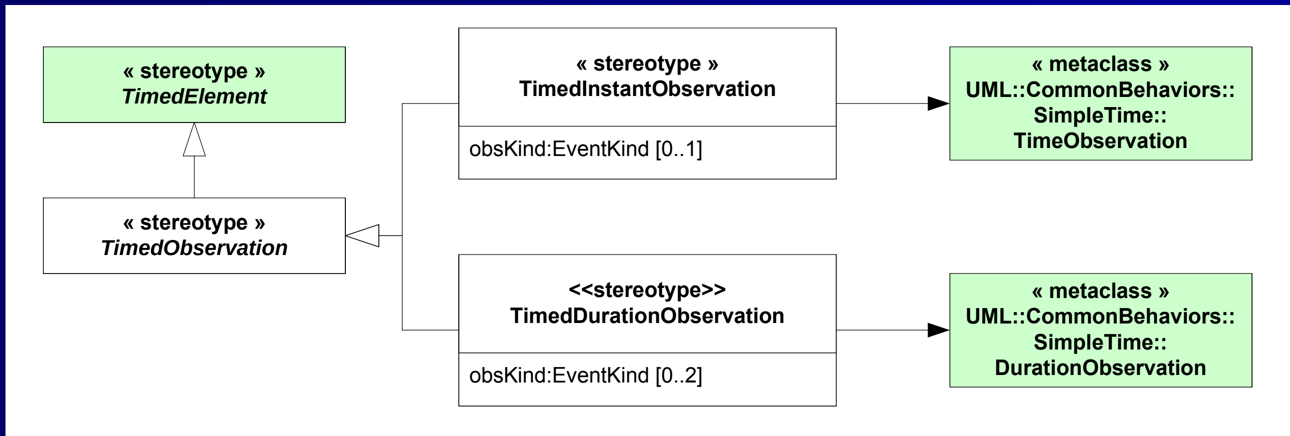


Extending the TimeEvent metaclass of SimpleTime

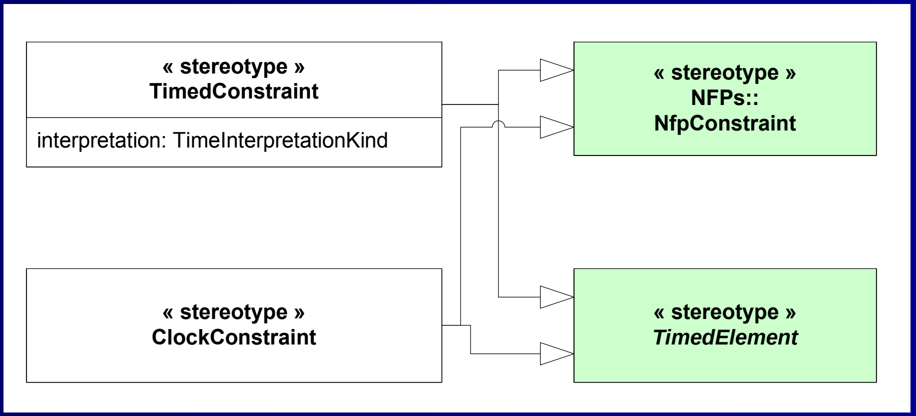
TimedProcessing



TimedObservation

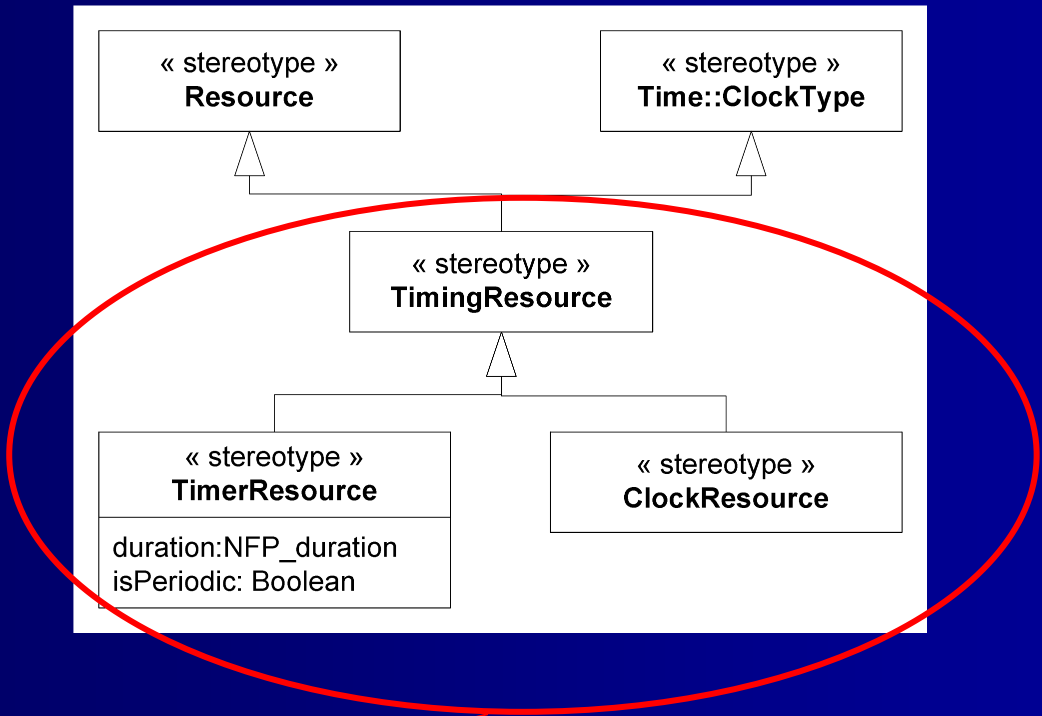


TimedConstraint & ClockConstraint



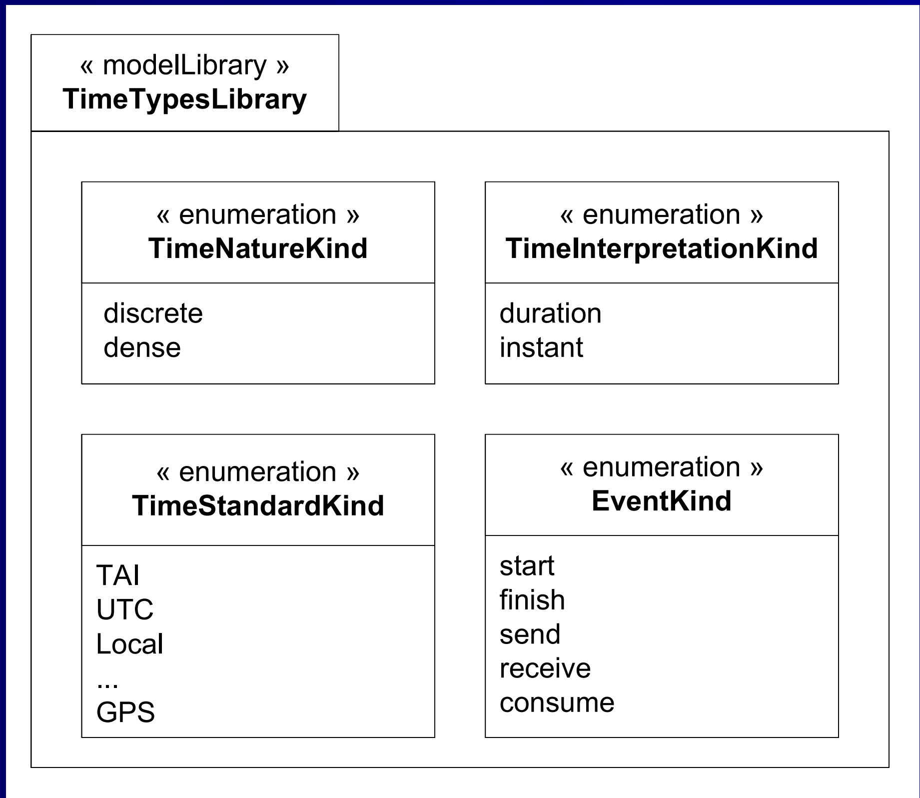
Time-related GRM stereotypes

Sterotypes defined in the Generic Resource Modeling sub-profile



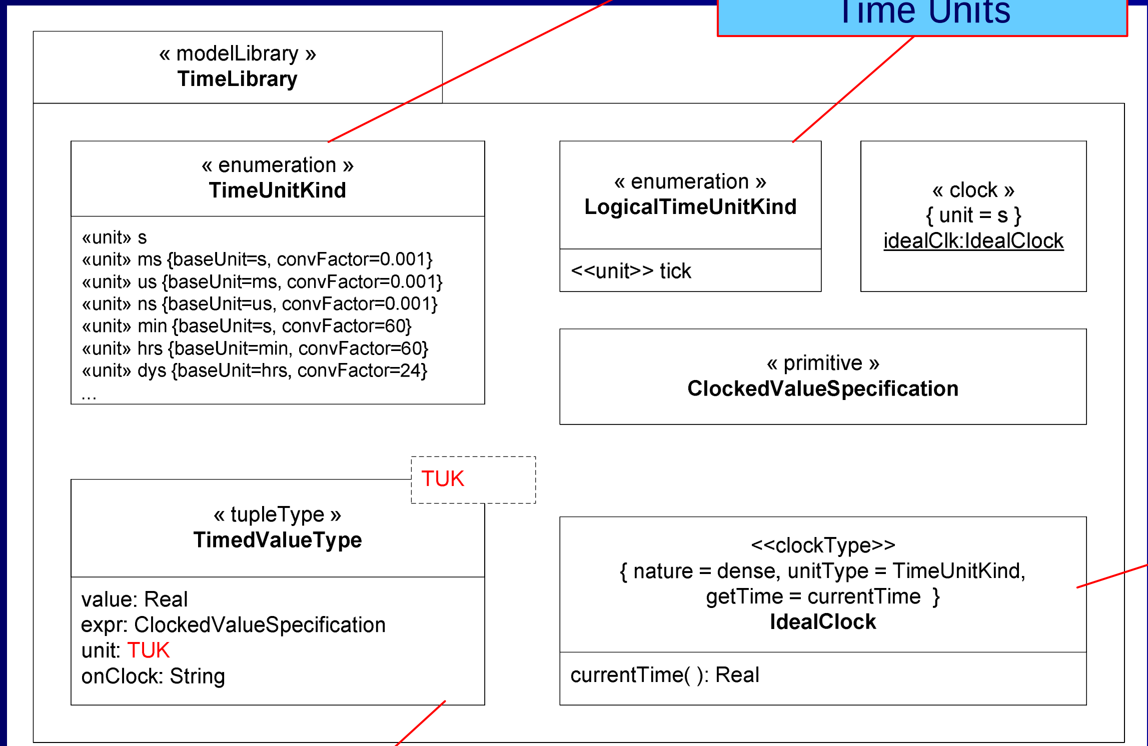
Resources for time management

Time-related libraries: TimeTypesLibrary



Time-related libraries: TimeLibrary

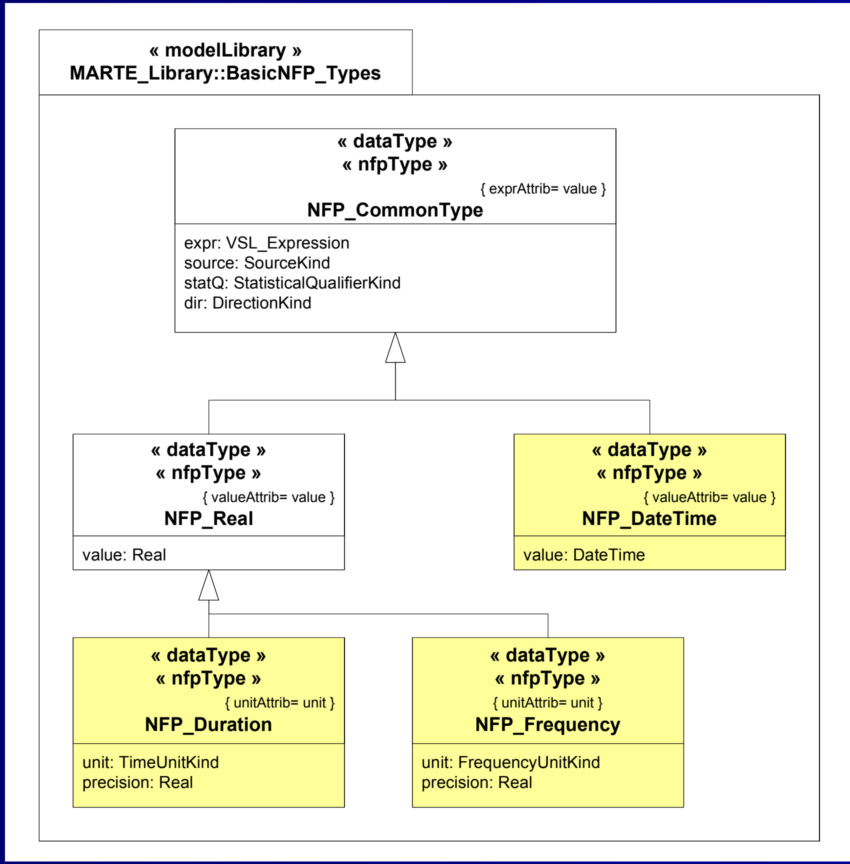
Two usual sets of Time Units



Model of ideal "physical time"

Templated DataType

Time-related NFP types



Time Values: Concrete syntax

- Simple time values
  (value=3.5, unit=ms, onClock='idealClk');
  3.5 ms on idealClk;
- Homogemeous expressions
  (value=1.5, unit=ms, onClock='idealClk') +
  (value=150, unit=us, onClock='idealClk');
  → (value=1650, unit=us, onClock='idealClk')
- Heterogeneous expressions
  min (15 tick on prClk, 5 ms on idealClk);
- Additional capabilities with VSL
  – Occurrence number, jitter,...
  – but implicitly on idealClk

tuple, *a la* VSL

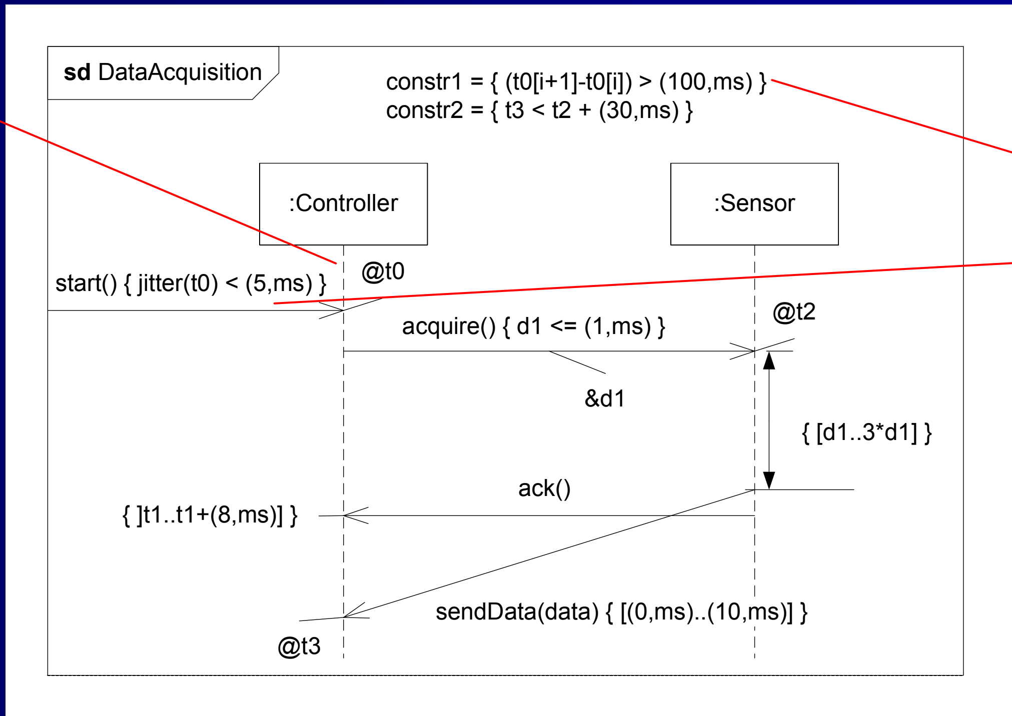
short form

Can be evaluated, because convFactor between units

Clock relation between prClk and idealClk must be provided

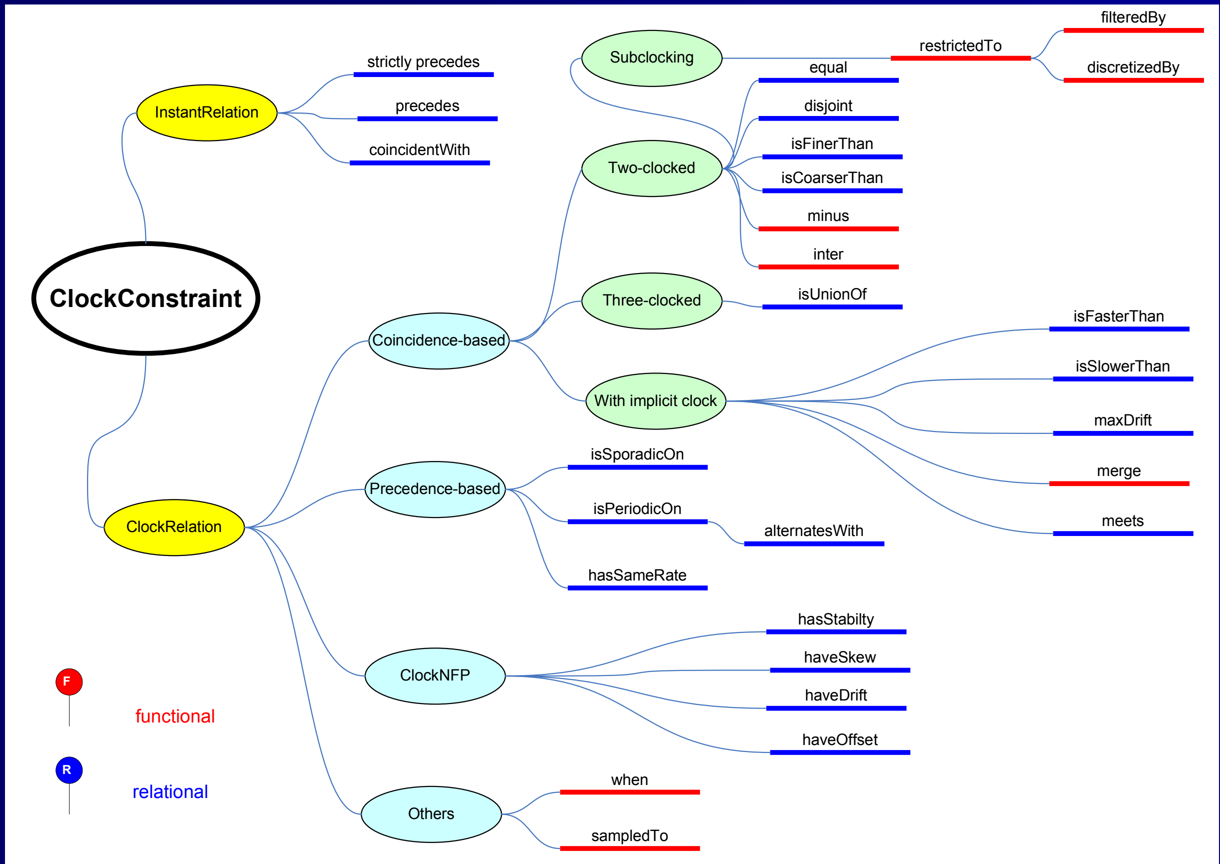
Time specific languages: VSL Time Constraints

t0: observation of the message: start



t0 is periodic, period 100ms with a jitter less than 5ms

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

Acknowledments

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

References

- [1] T. Henzinger, J. Sifakis. "*The embedded systems design challenge*". Proceedings of the 14th International Symposium on Formal Methods (FM), Lecture Notes in Computer Science, Springer, August, 2006.
- [2] B. Hailpern, P. Tarr. *"Model-driven development: The good, the bad, and the ugly".* IBM SYSTEMS JOURNAL, VOL 45, NO 3, 2006, pp 451-461.
- [3] A. Sangiovanni-Vincentelli. *"Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design".* Proceedings of the IEEE, Vol. 95, No. 3, March 2007, pp 467-506.
- [4] S. Gérard, J. Medina, D. Petriu. *"MARTE: A New Standard for Modeling and Analysis of Real-Time and Embedded Systems".* Tutorial, ECRTS 2007, Pisa (I), July 4, 2007
- [5] B. Selic. *"A systematic approach to domain-specific language design using UML".* In *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*, pages 2–9. IEEE, 2007.