Agenda

- Part 1
  - Introduction to MDD for RT/E systems & MARTE in a nutshell
- Part 2
  - Non-functional properties modeling
  - Outline of the Value Specification Language (VSL)
- Part 3
  - **The timing model**
- Part 4
  - A component model for RT/E
- Part 5
  - Platform modeling
- Part 6
  - Repetitive structure modeling
- Part 7
  - Model-based analysis for RT/E
- Part 8
  - MARTE and AADL
- Part 9
  - Conclusions

Time in MARTE Overview

- **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
- **Usage of the Time sub-profile**

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

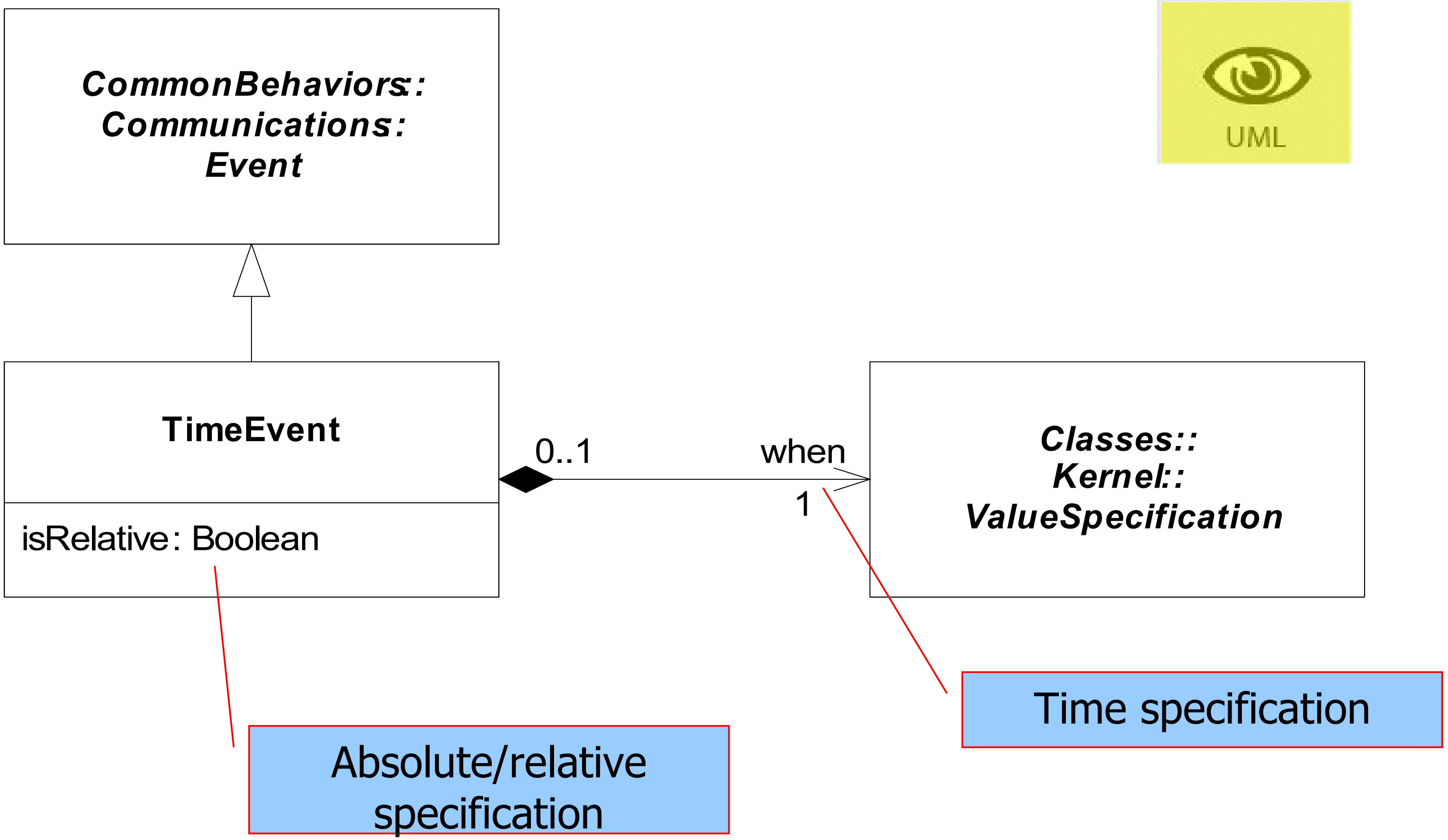
- UML2 adds new metaclasses to represent
  - Time
  - Duration
  - Observation (of time passing)
  - Some forms of time constraints

- 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

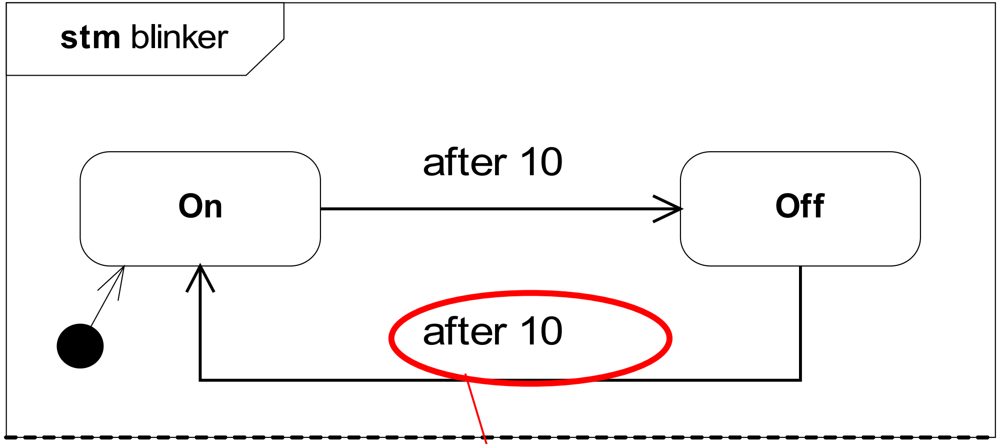


Absolute/relative specification

Time specification

TimeEvent – usage (1)

UML state machine = behavior



User

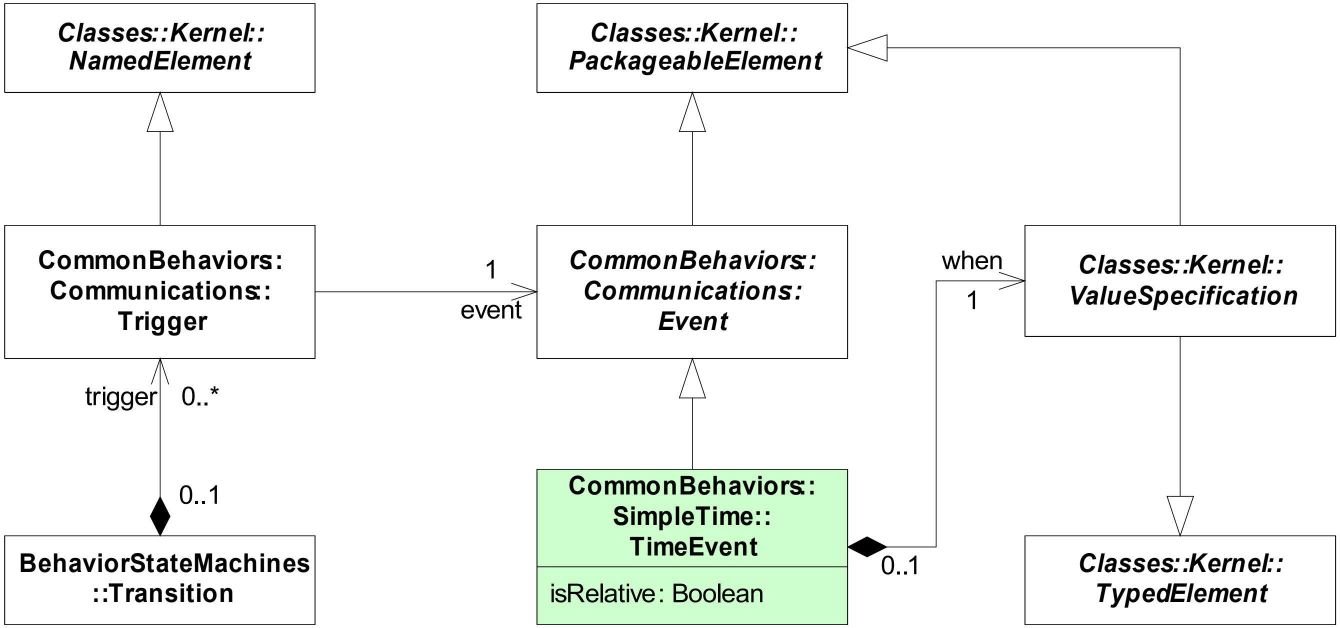
Specification of a time-trigger

Informal semantics

TimeEvent – usage (2)

UML

Meaning of “after 10”

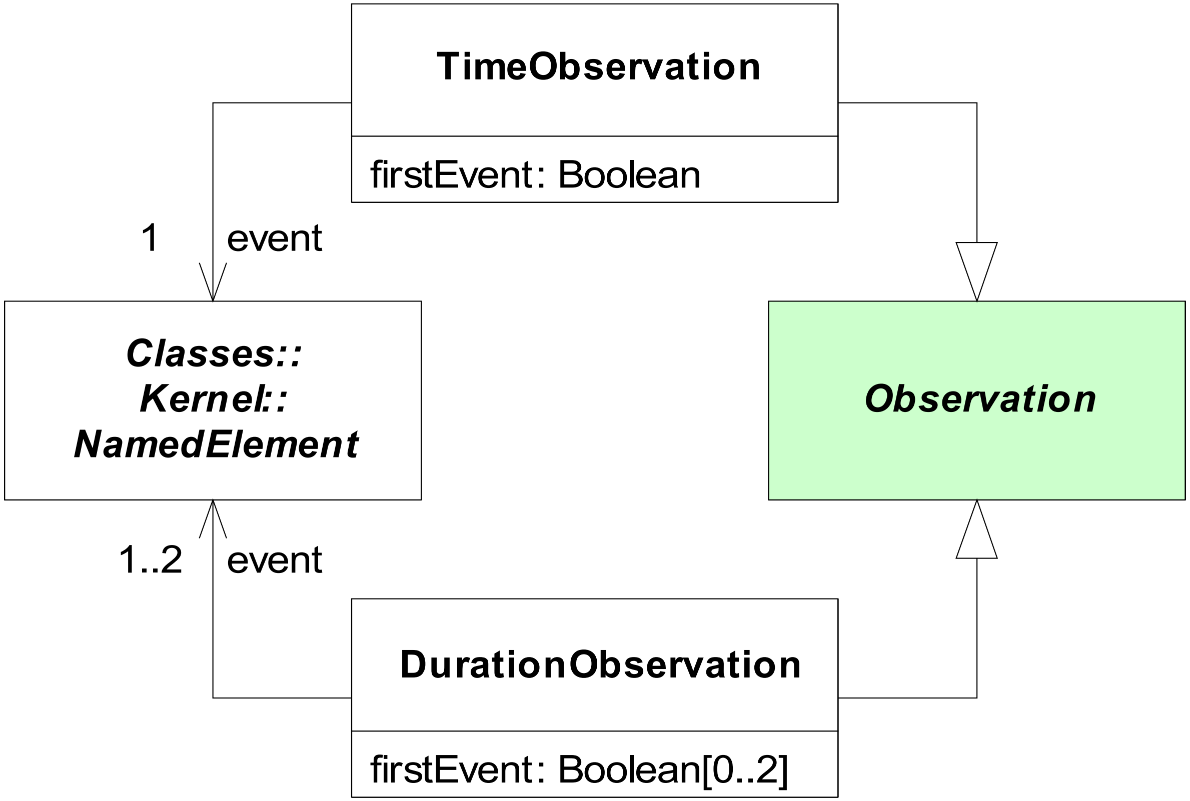


Metaclasses involved in the modeling of a transition triggered by a TimeEvent

Simple annotation →
complex implied structure

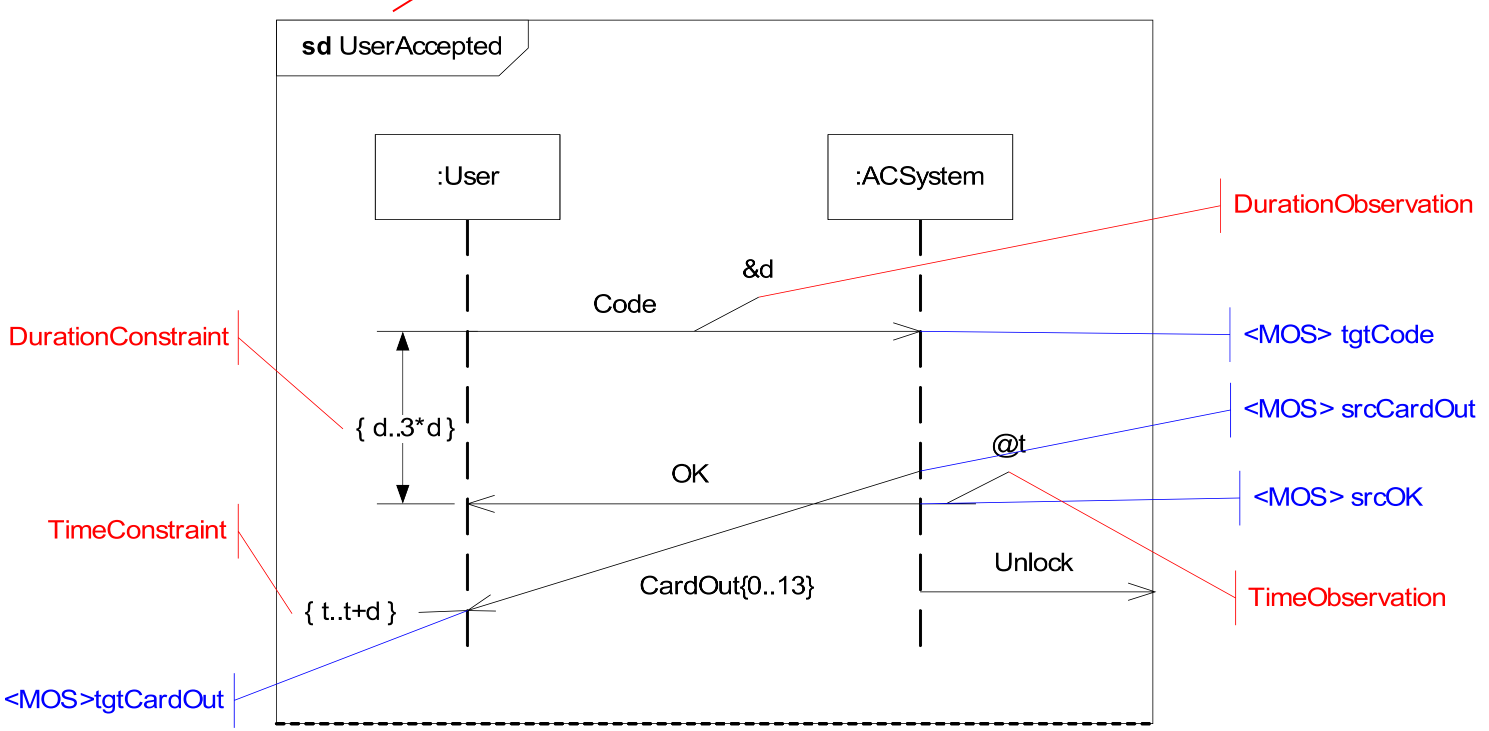
SimpleTime::Observation

UML



Observation – usage (1)

Example of sequence diagram



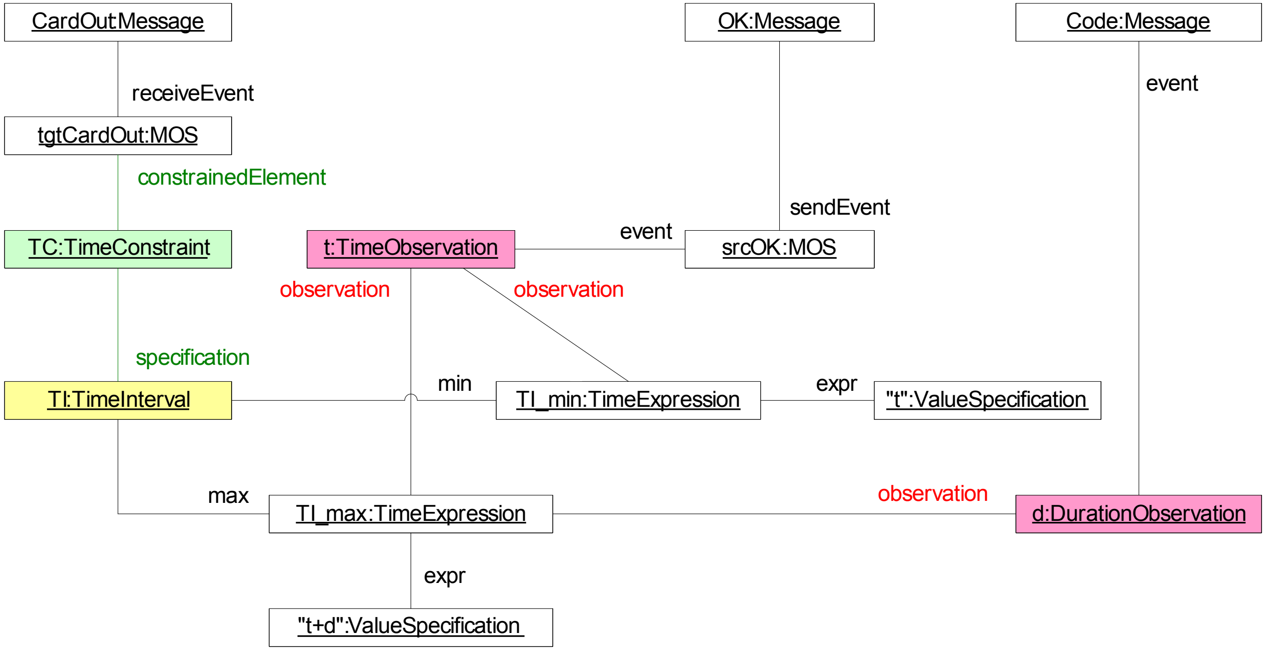
User

MOS stands for MessageOccurrenceSpecification

Note that red and blue annotations are not part of the UML notation.

Observation – usage (2)

Instance model of the time constraint: receive CardOut in {t .. t+d}



Simple annotation → complex implied structure

Overview

- 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
- Usage of the Time sub-profile

Domain

Concepts in MARTE::Time (1/2)

- **Time structure =**

  set of time bases + time structure relations

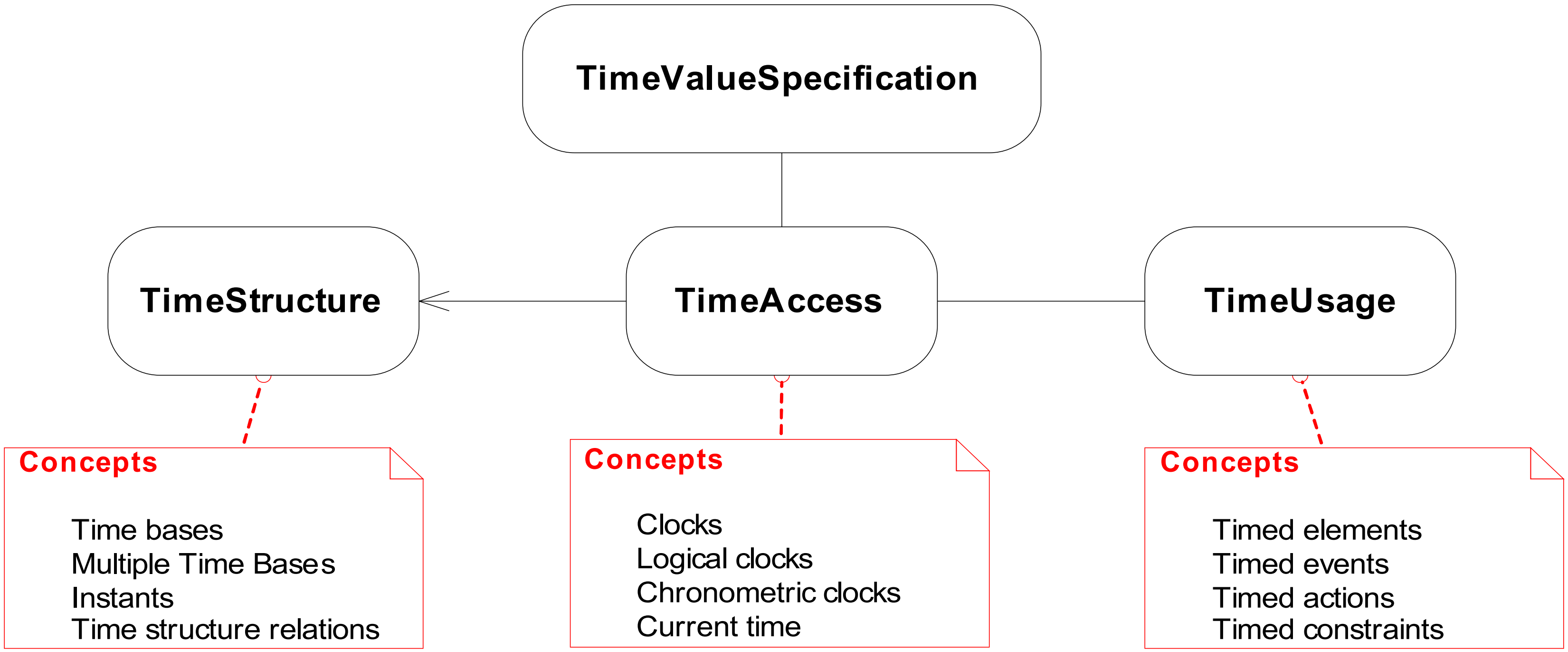
  → Partially ordered set of instants

- **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 (2/2)

Main concepts introduced in Time modeling


TimeValueSpecification

TimeStructure

TimeAccess

TimeUsage

Concepts

Time bases
Multiple Time Bases
Instants
Time structure relations

Concepts

Clocks
Logical clocks
Chronometric clocks
Current time

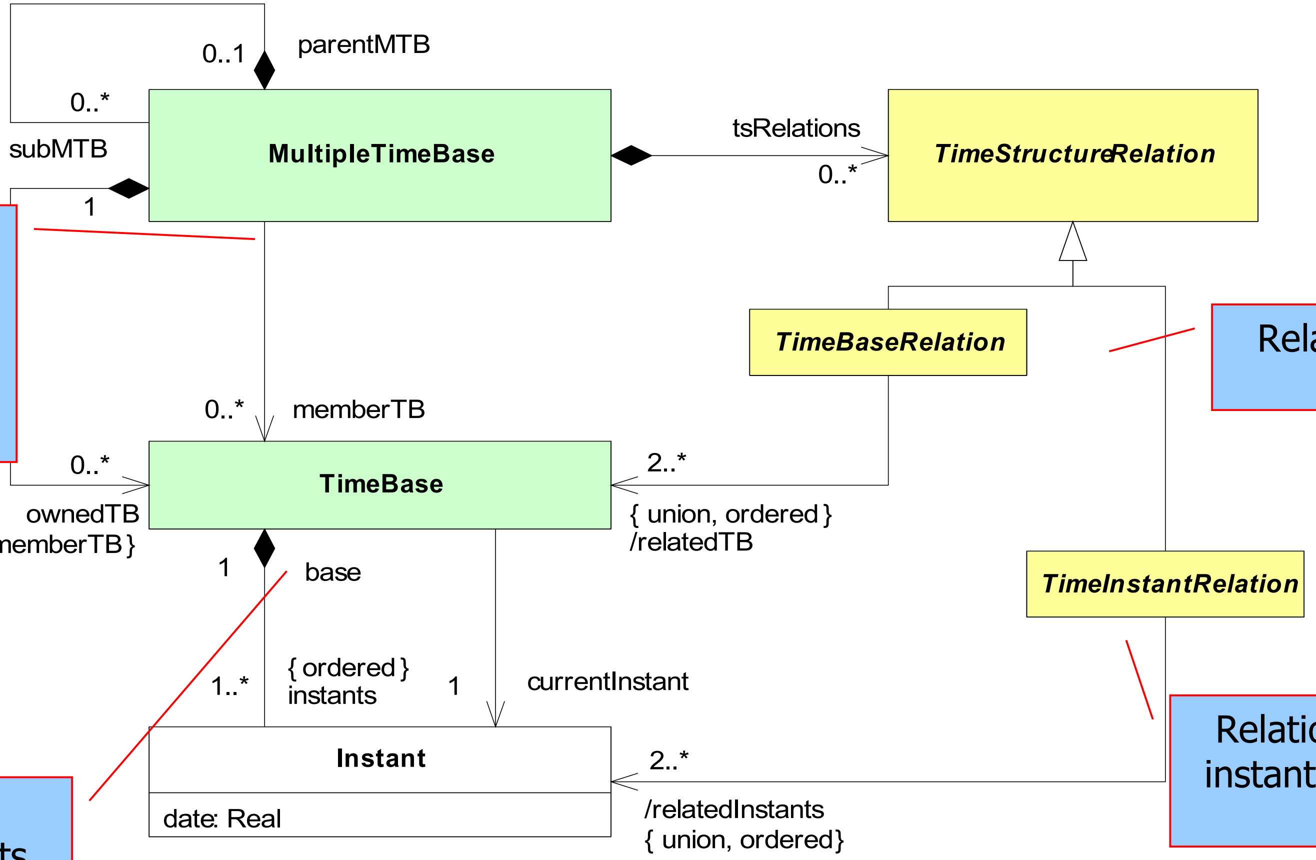
Concepts

Timed elements
Timed events
Timed actions
Timed constraints

Not a UML diagram!

Time Structure

Domain



MultipleTimeBase = set of TimeBases + Hierarchy + Constraints

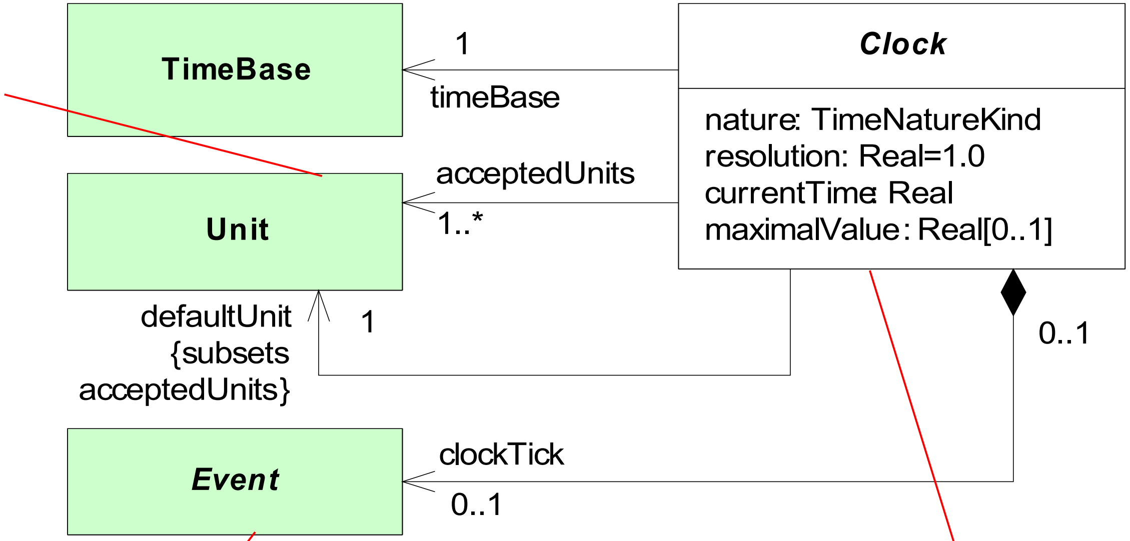
Relationships over TBs

Relationships over instants of different TBs

TimeBase = oset of instants

Access to Time: Clock

Domain



Units associated with a clock

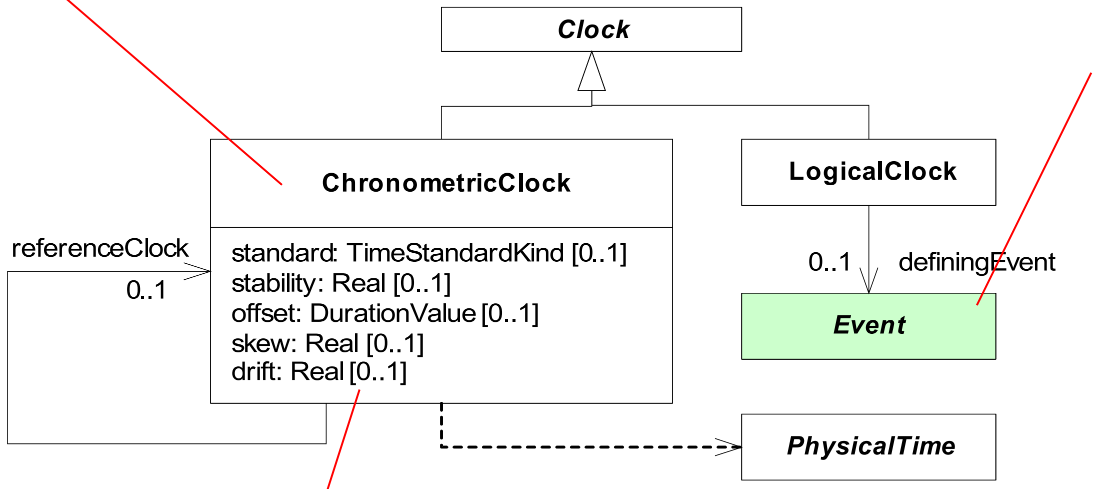
Access to the time structure

Event occurring at each clock ticking

Chronometric/Logical Clocks

Two kinds of clocks

Domain



Implicit reference to physical time

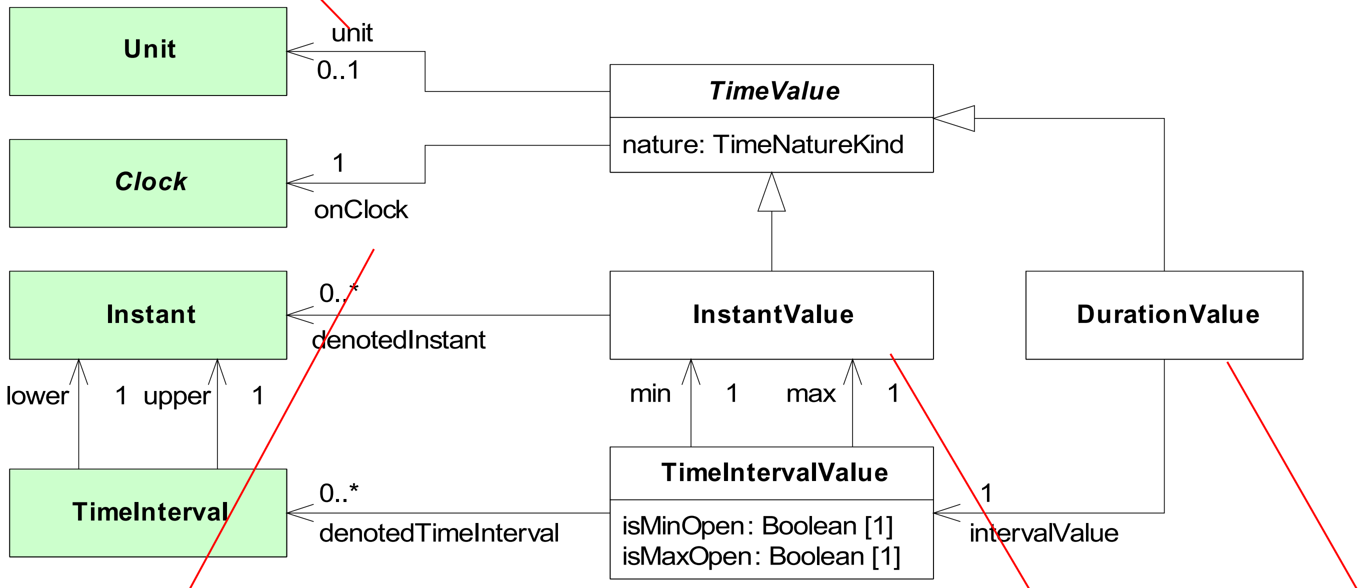
Possible reference to a repetitive event

NFPs measured against a reference clock

Time Values

Domain

A TimeValue has a unit (default= clock unit)



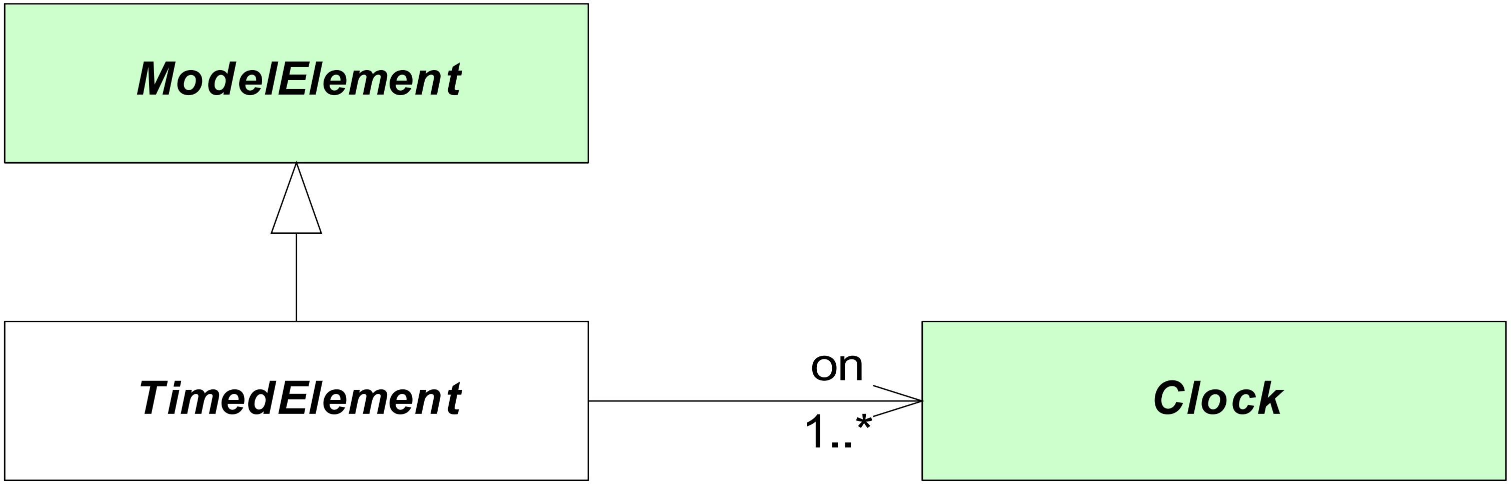
A TimeValue must reference a clock

Instant/Duration two distinct concepts

Timed Entities: TimedElement

Domain

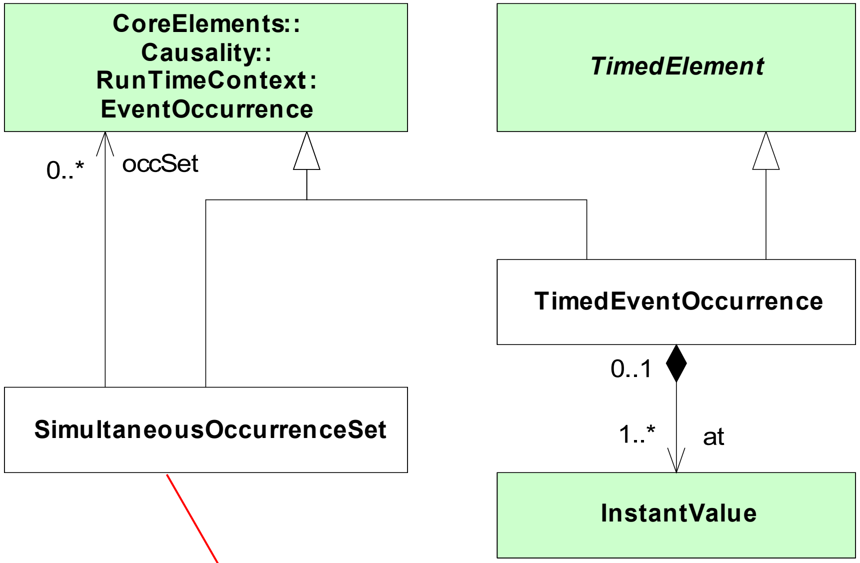
The unifying concept: a TimedElement = a ModelElement + a Clock



Timed Entities: TimedEvent

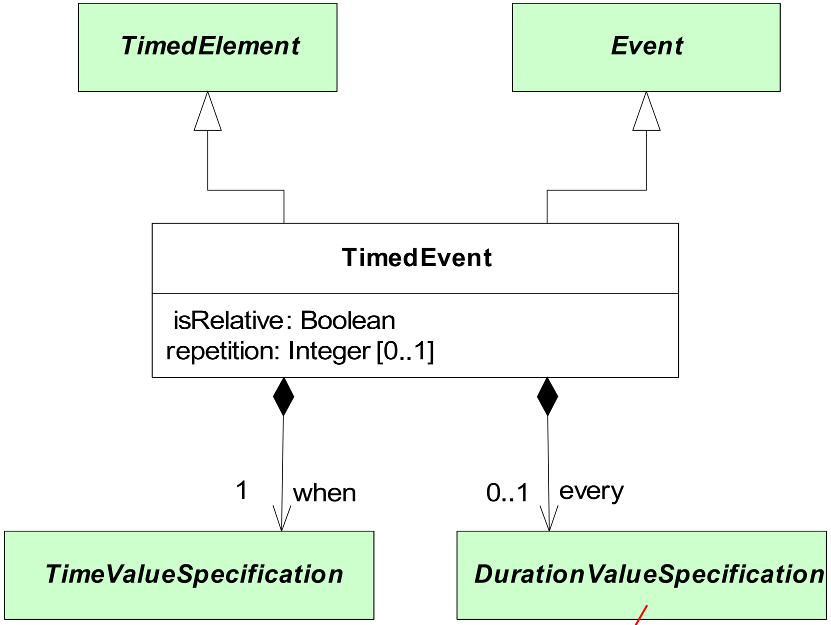
Domain

occurrences



Provision for simultaneity

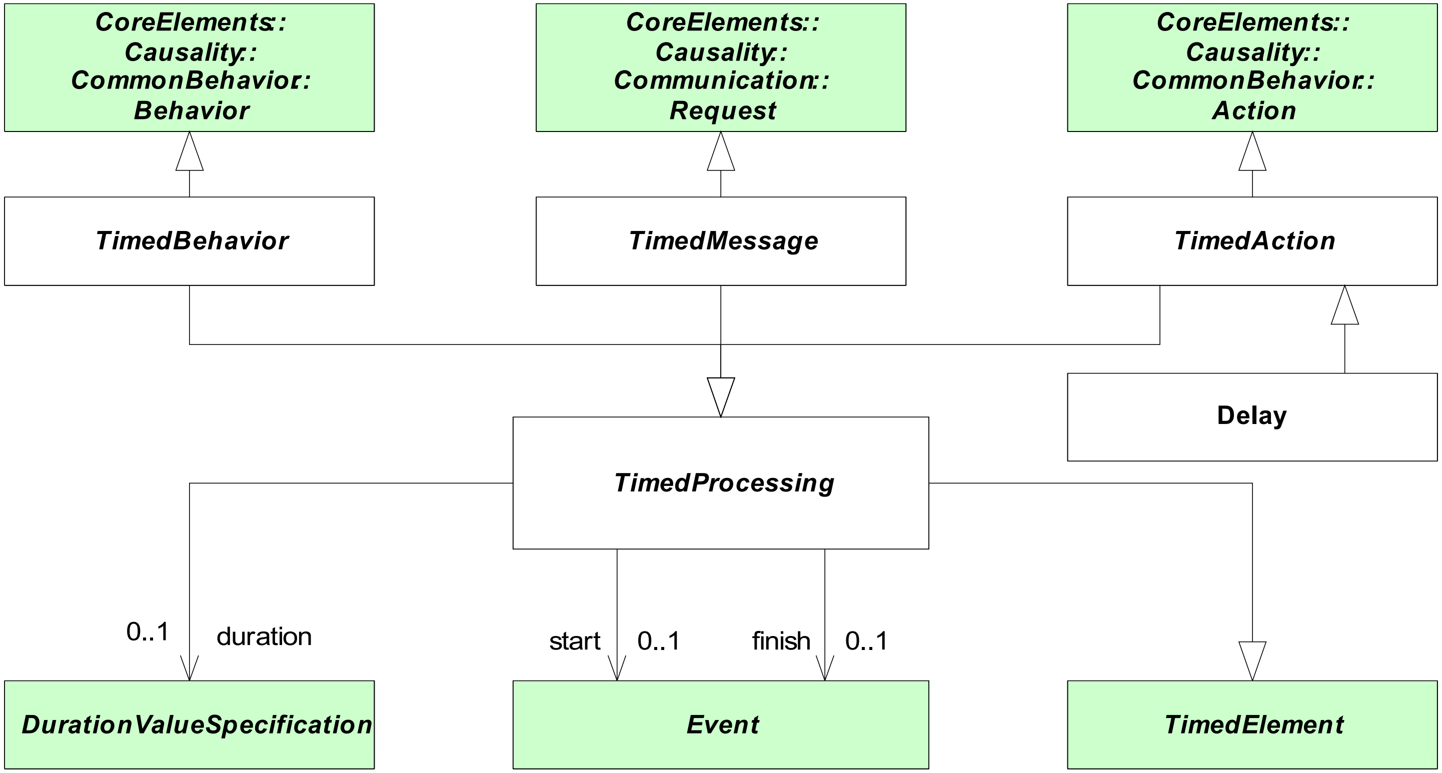
events



Facility to specify multiple occurrences

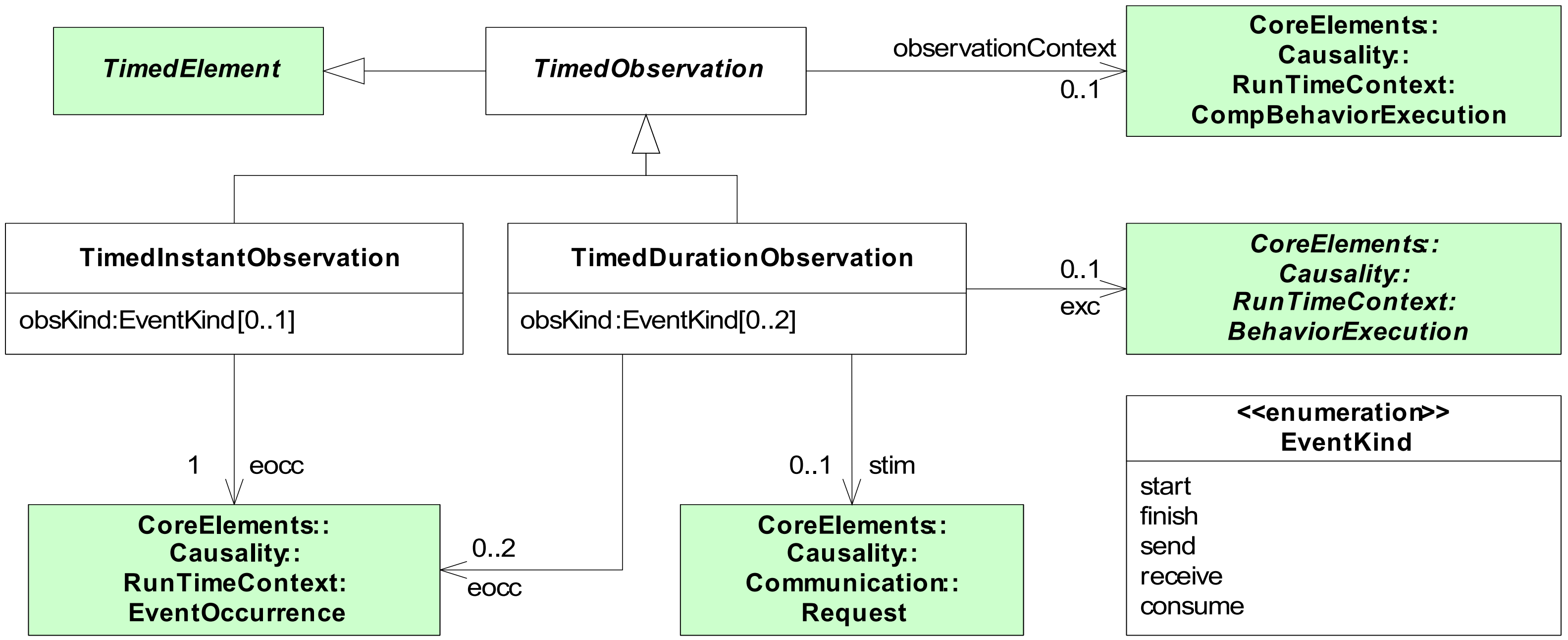
Timed Entities: TimedProcessing

Domain



Timed Entities:TimedObservation

Domain



Artistic

See:

http://en.wikipedia.org/wiki/Image:The_Persistence_of_Memory.jpg

Overview

- 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**
- Usage of the Time sub-profile

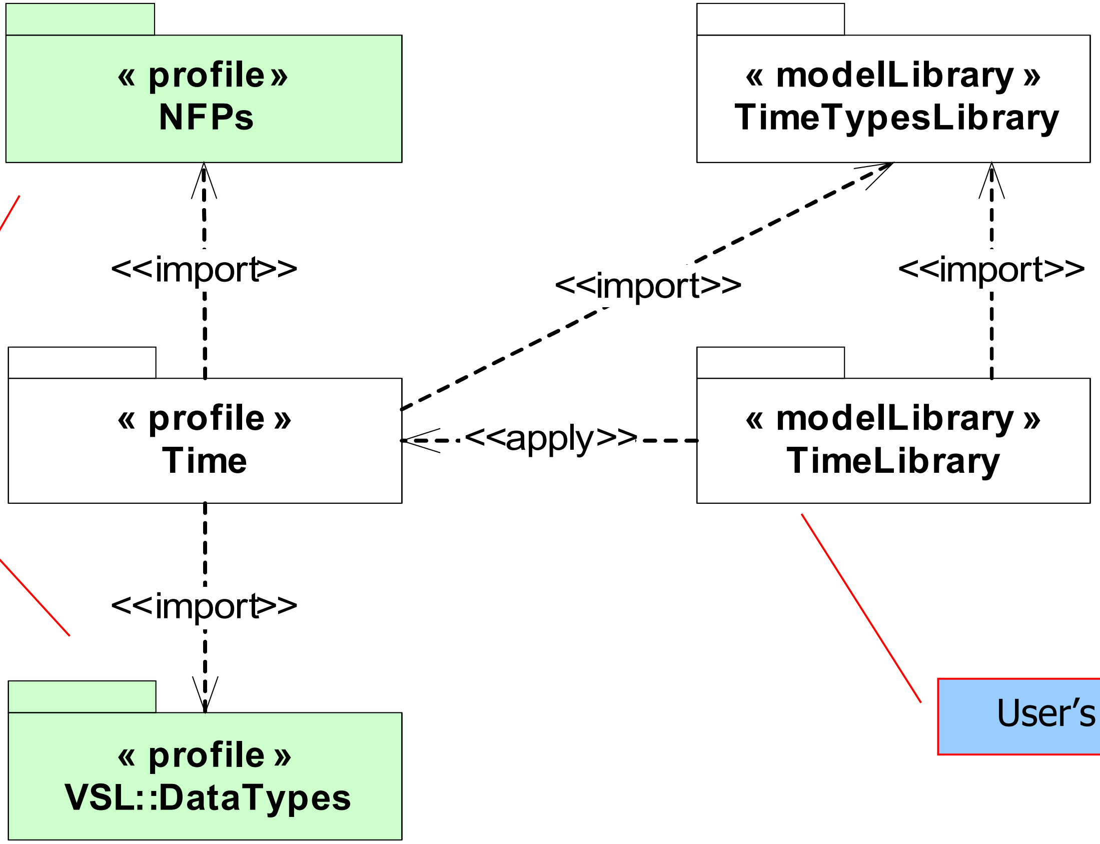
UML

Providing extensions to UML

- Through a UML profile
  - New Stereotypes
- Facilities
  - Model libraries
  - Dedicated languages (especially for expressions)

Dependencies

UML



Two other sub-profiles 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**

Stereotype properties :
Special semantics

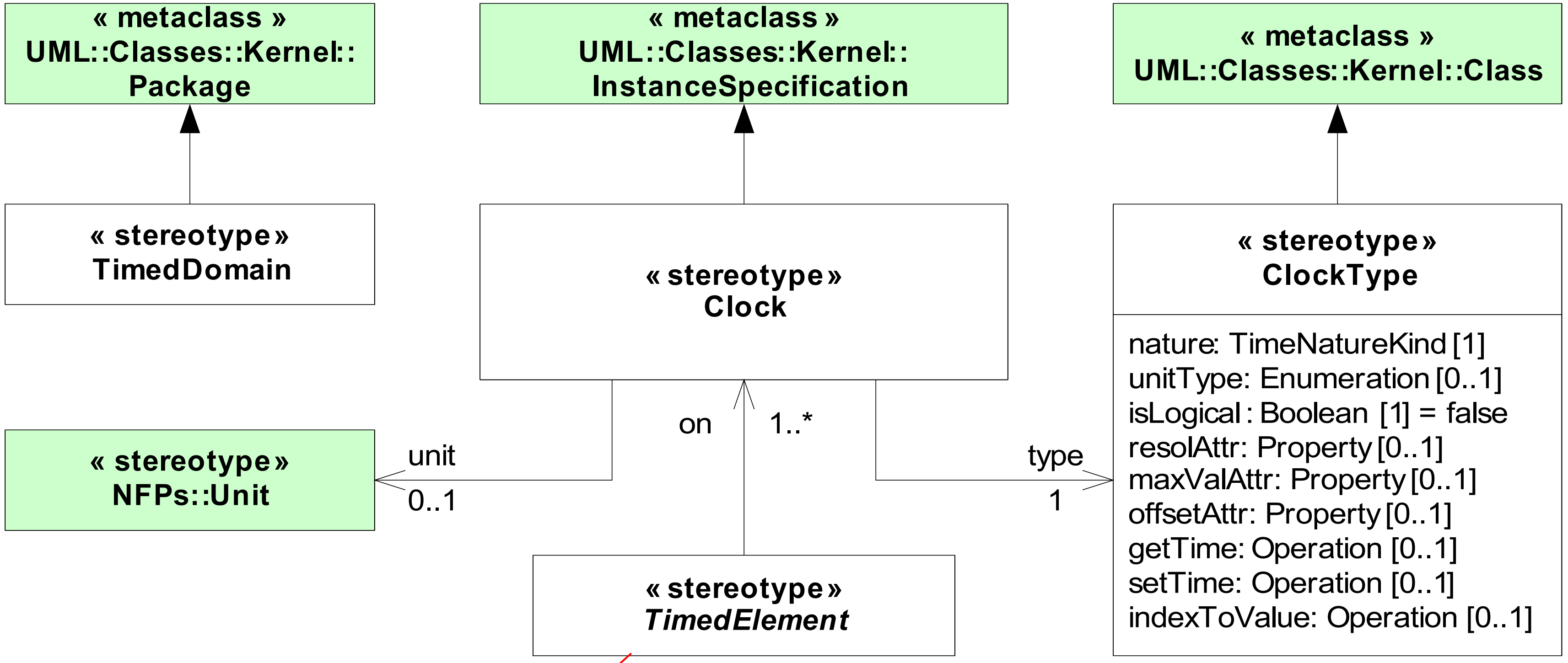
+ optional

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

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

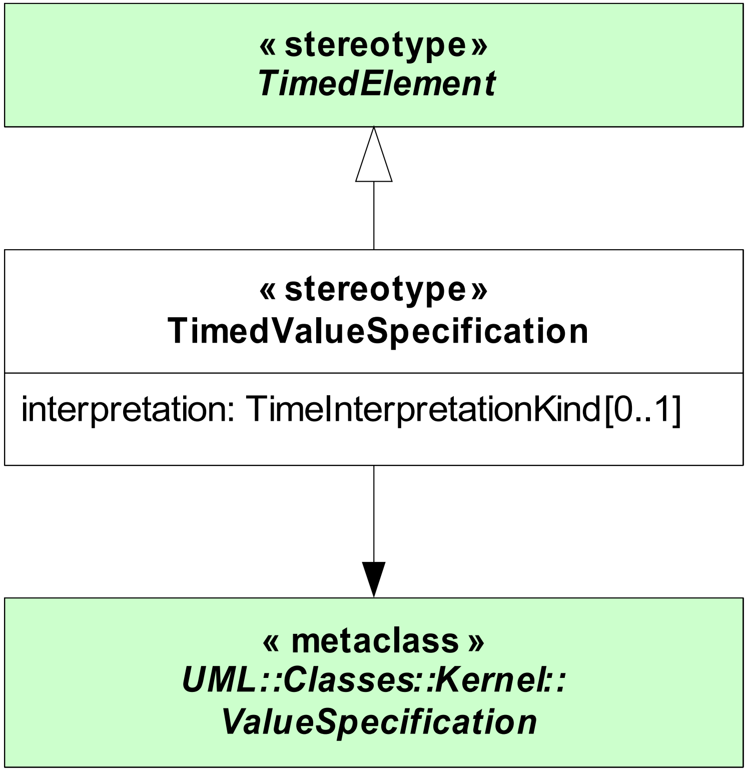
Clock and TimedElement

UML


« metaclass »
UML::Classes::Kernel::Package
« metaclass »
UML::Classes::Kernel::InstanceSpecification
« metaclass »
UML::Classes::Kernel::Class
« stereotype »
TimedDomain
« stereotype »
Clock
« 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]
« stereotype »
NFPs::Unit
unit
0..1
on
1..*
type
1
« stereotype »
TimedElement

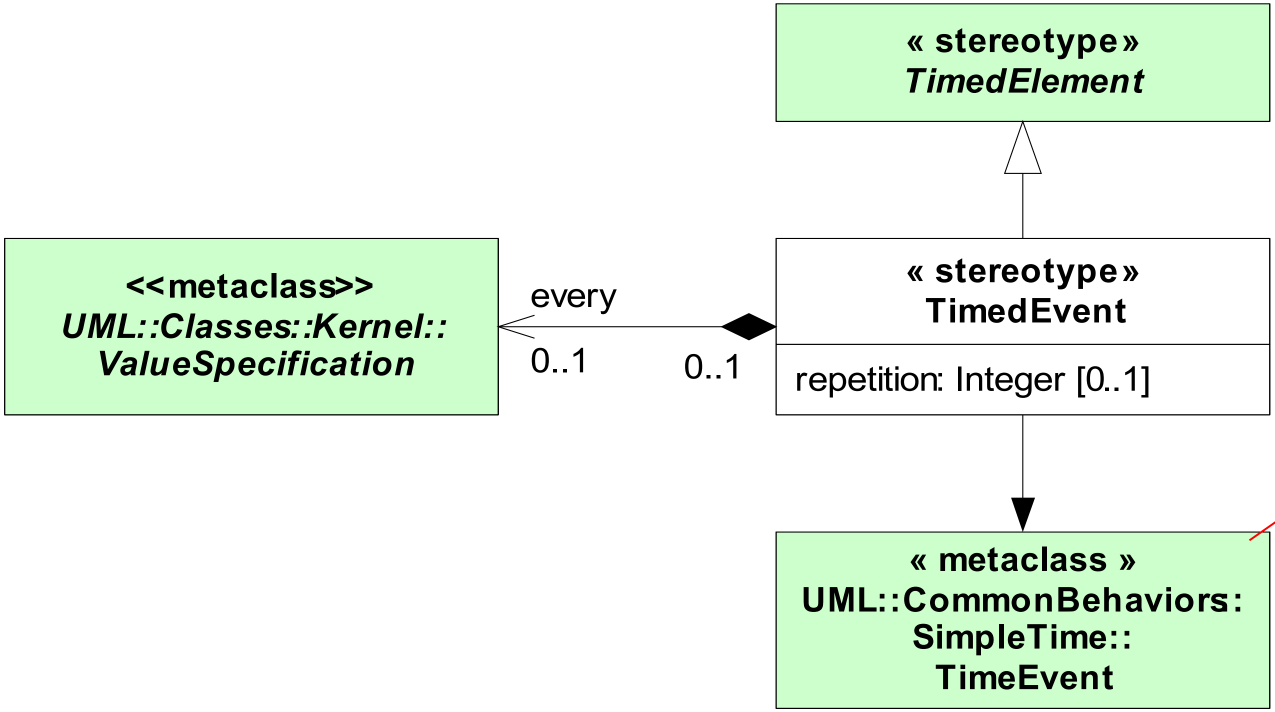
Notice that this abstract stereotype has no base metaclass

TimedValueSpecification



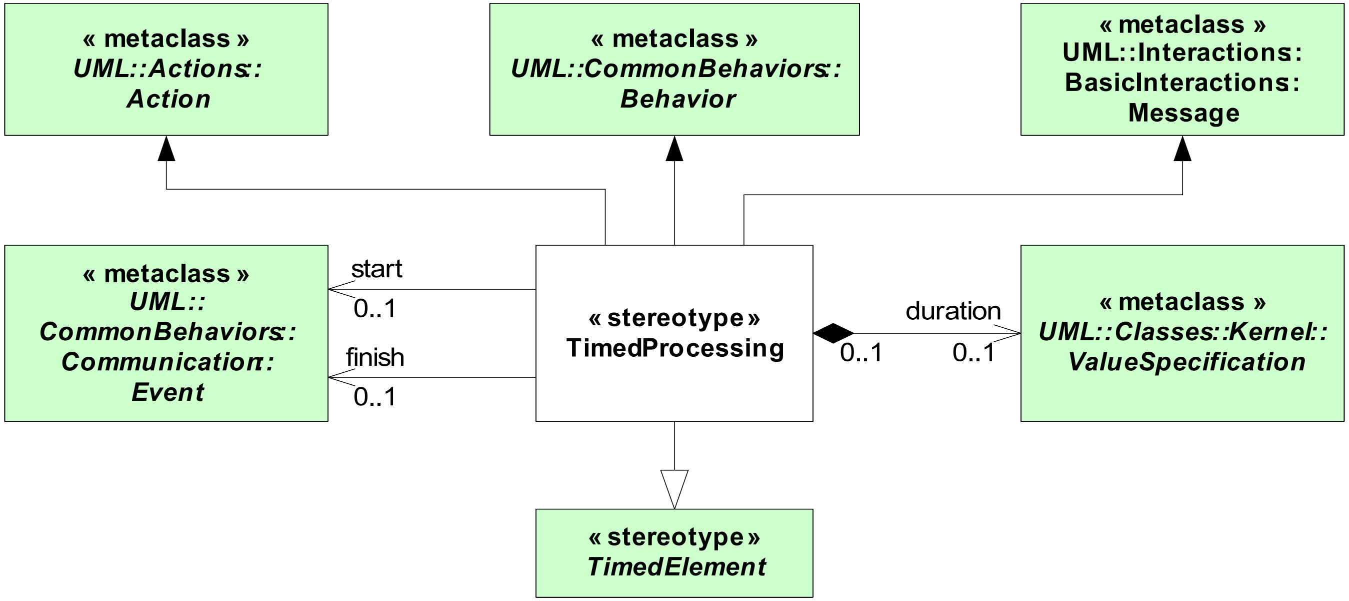
- either Instant
- or Duration

TimedEvent



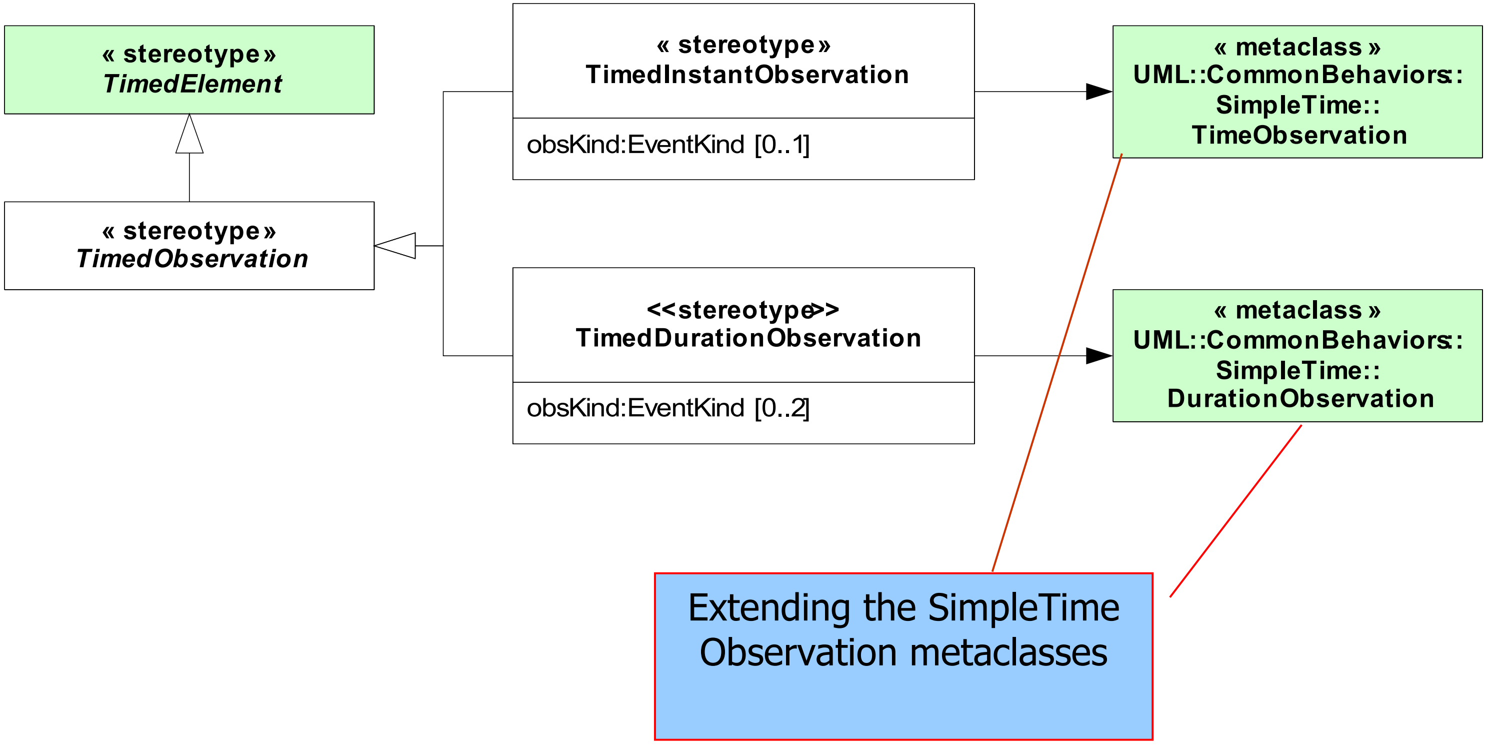
Extending the TimeEvent metaclass of SimpleTime

TimedProcessing



TimedObservation

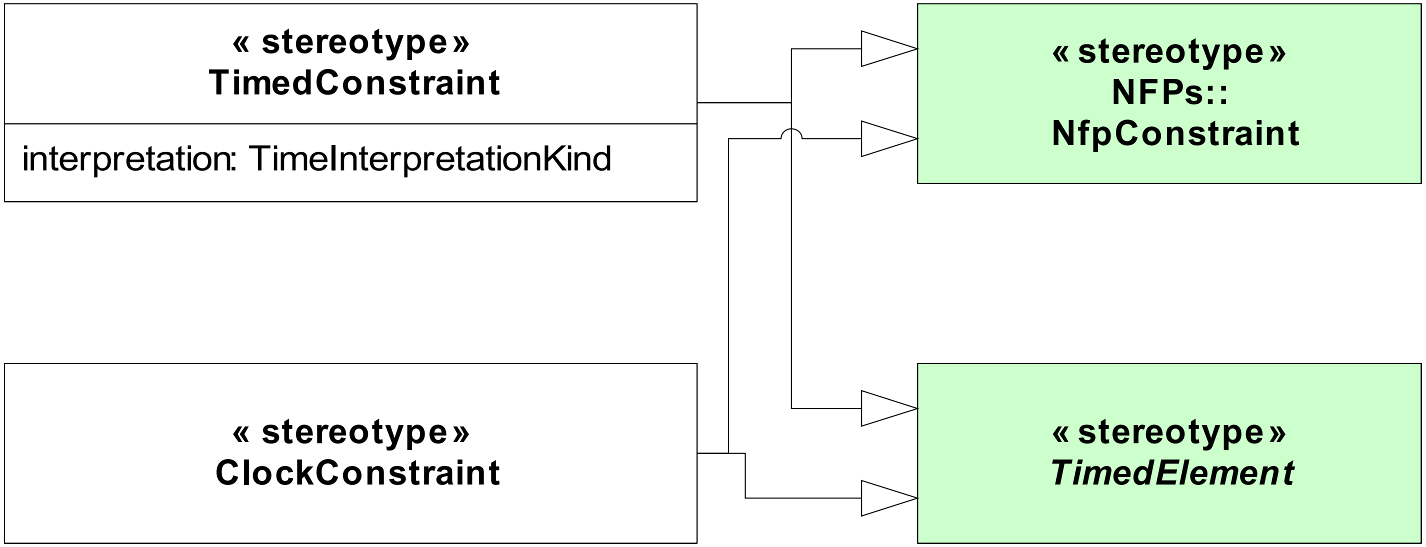
UML



Extending the SimpleTime Observation metaclasses

TimedConstraint & ClockConstraint

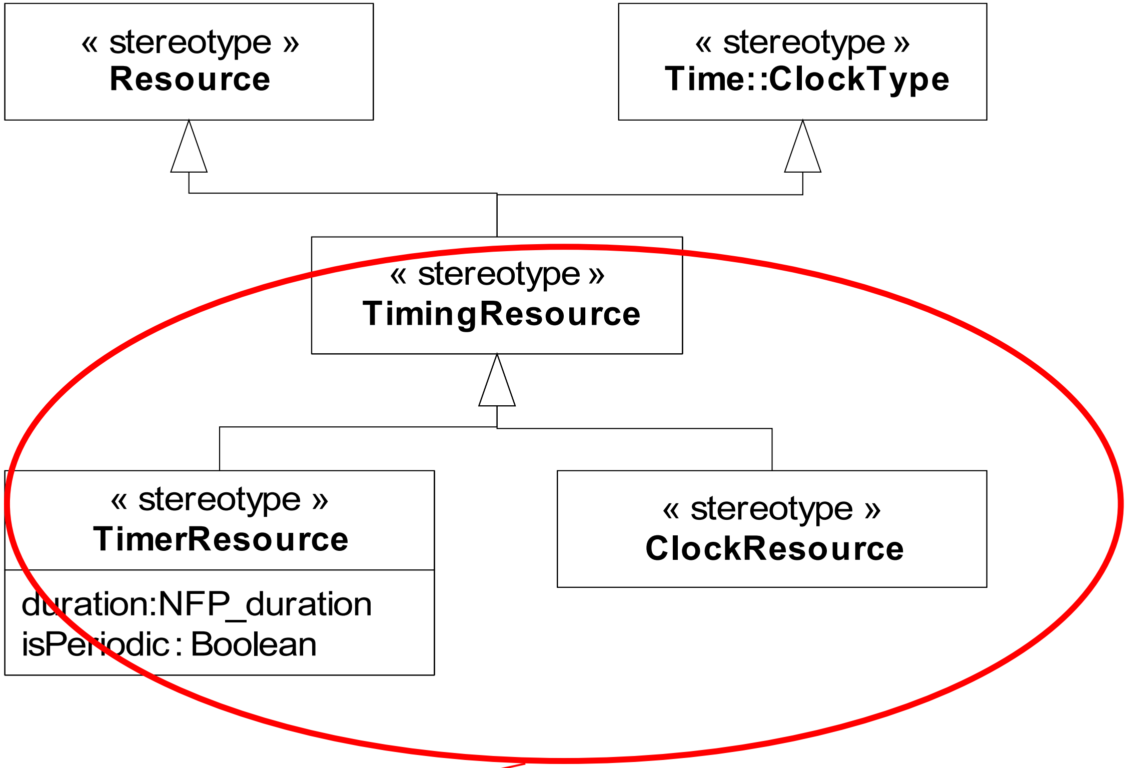
UML



Time-related GRM stereotypes

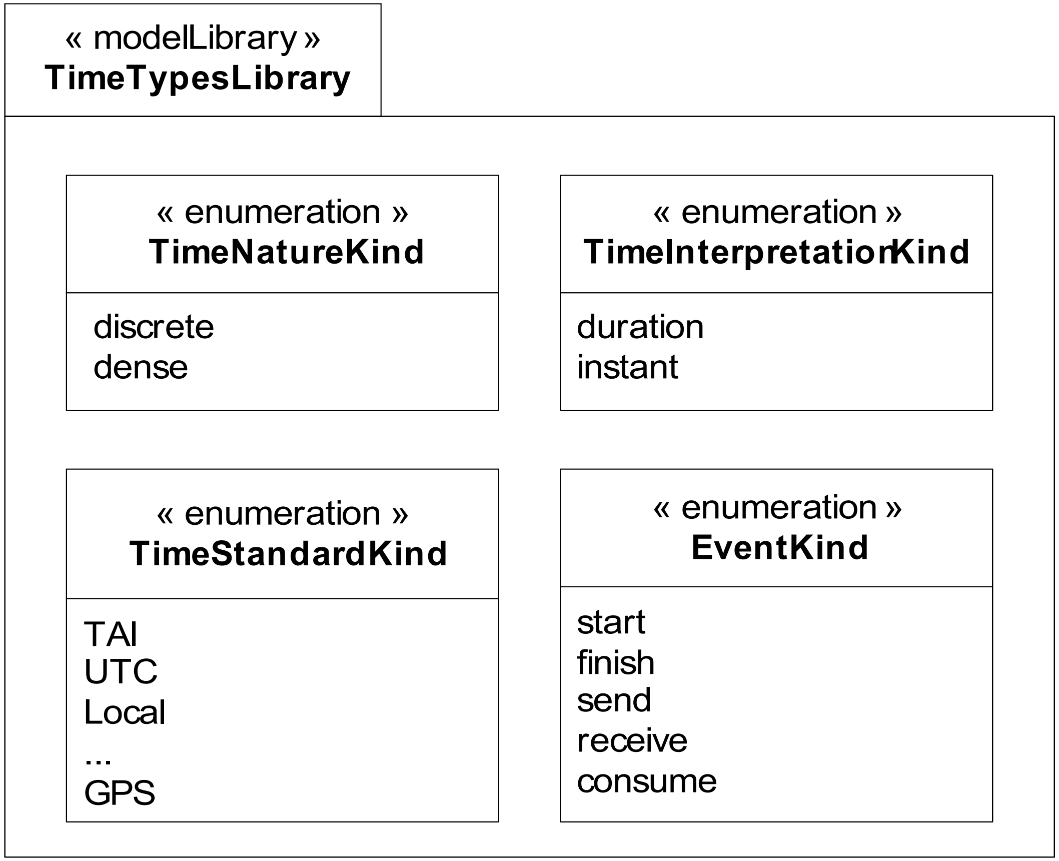
Sterotypes defined in the Generic Resource Modeling sub-profile

UML



Resources for time management

Time-related libraries: TimeTypesLibrary



« modelLibrary »
TimeTypesLibrary

« enumeration »
TimeNatureKind

discrete
dense

« enumeration »
TimeInterpretatiorKind

duration
instant

« enumeration »
TimeStandardKind

TAI
UTC
Local
...
GPS

« enumeration »
EventKind

start
finish
send
receive
consume

UML

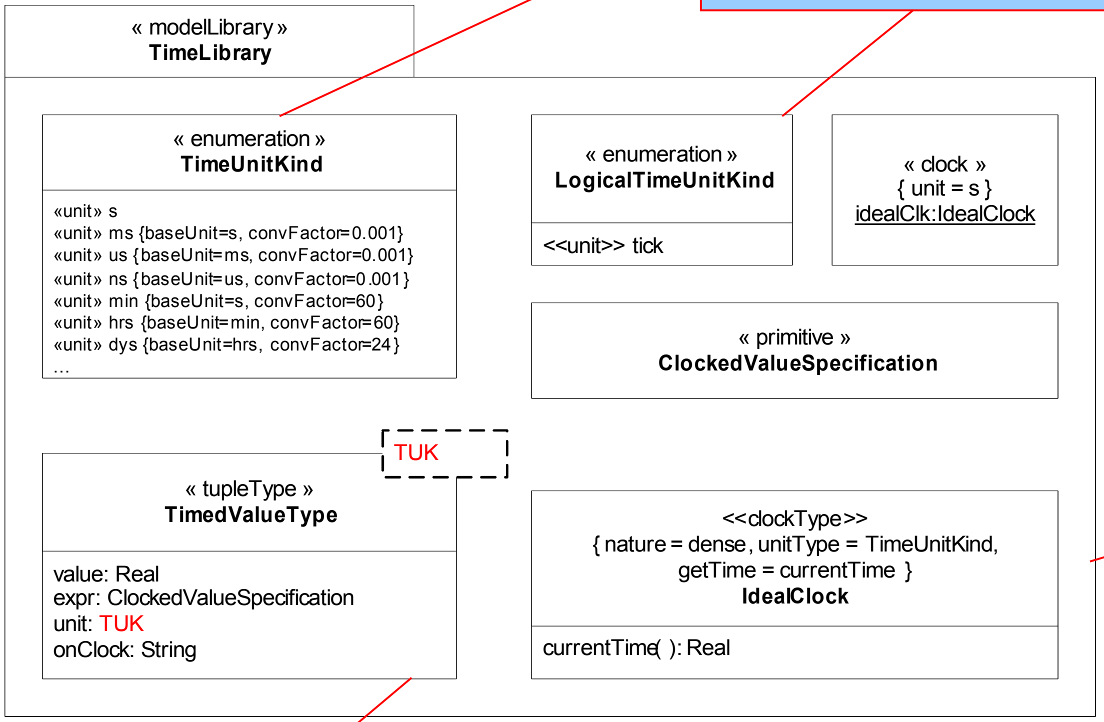
&

User

Time-related libraries: TimeLibrary

User

Two usual sets of Time Units



« modelLibrary »
TimeLibrary

« enumeration »
TimeUnitKind

«unit» s
«unit» ms {baseUnit=s, convFactor=0.001}
«unit» us {baseUnit=ms, convFactor=0.001}
«unit» ns {baseUnit=us, convFactor=0.001}
«unit» min {baseUnit=s, convFactor=60}
«unit» hrs {baseUnit=min, convFactor=60}
«unit» dys {baseUnit=hrs, convFactor=24}
…

« enumeration »
LogicalTimeUnitKind

<<unit>> tick

« clock »
{ unit = s }
idealClk:IdealClock

« primitive »
ClockedValueSpecification

TUK

« tupleType »
TimedValueType

value: Real
expr: ClockedValueSpecification
unit: TUK
onClock: String

<<clockType>>
{ nature = dense, unitType = TimeUnitKind,
getTime = currentTime }
IdealClock

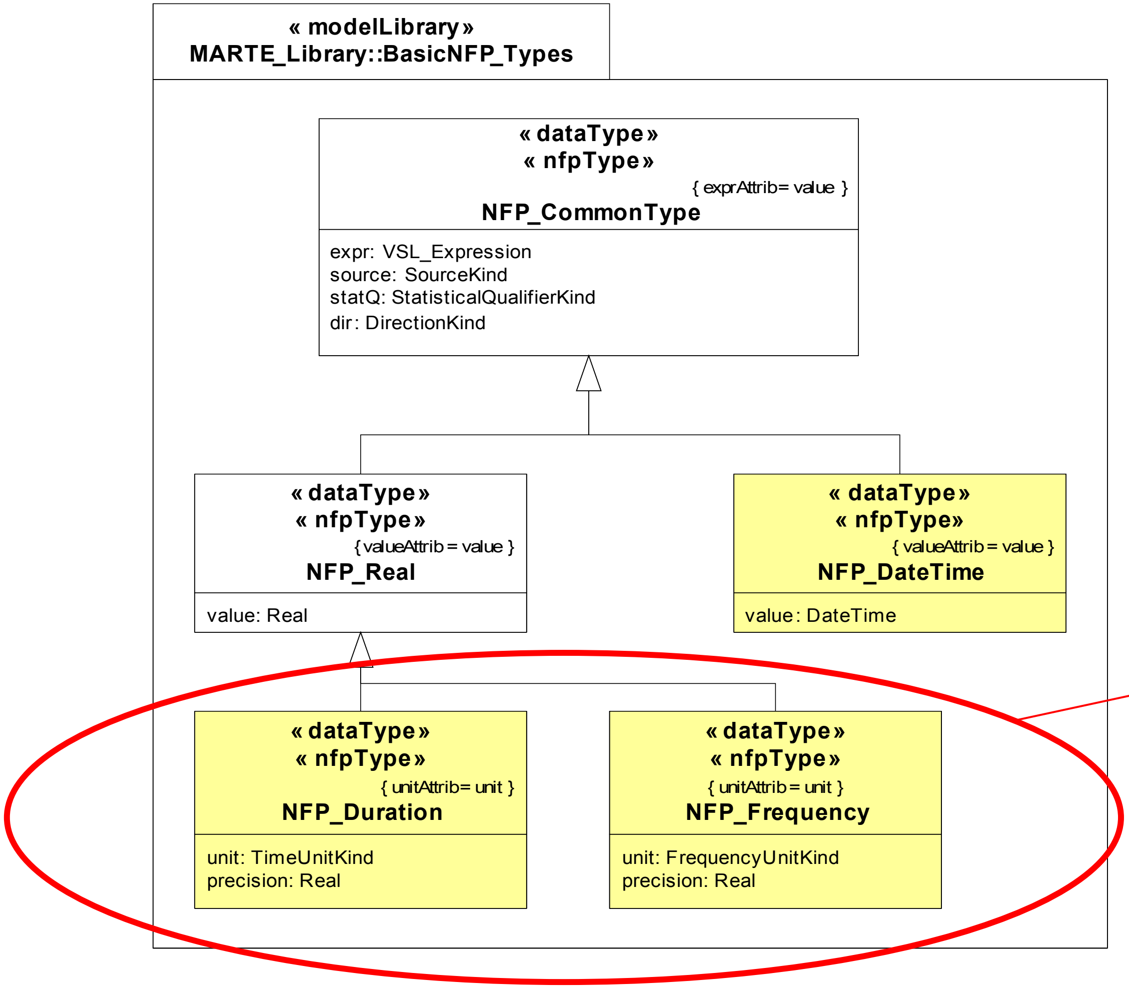
currentTime( ): Real

Model of ideal “physical time”

Templated DataType

Time-related NFP types

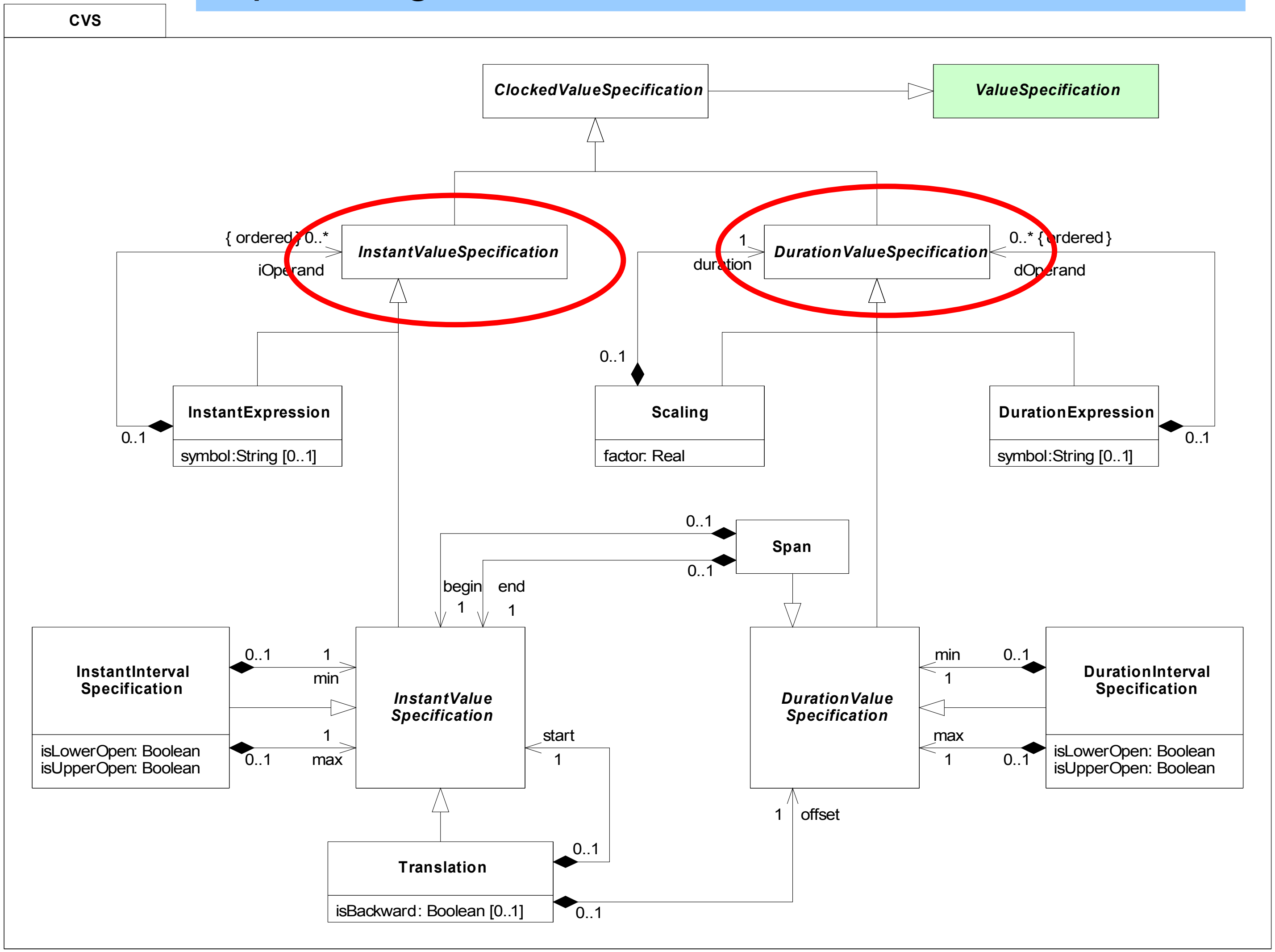
User



Time-related types.
Often used.

Time specific languages:Clocked Value Specification

Expressing time values with EXPLICIT clocks

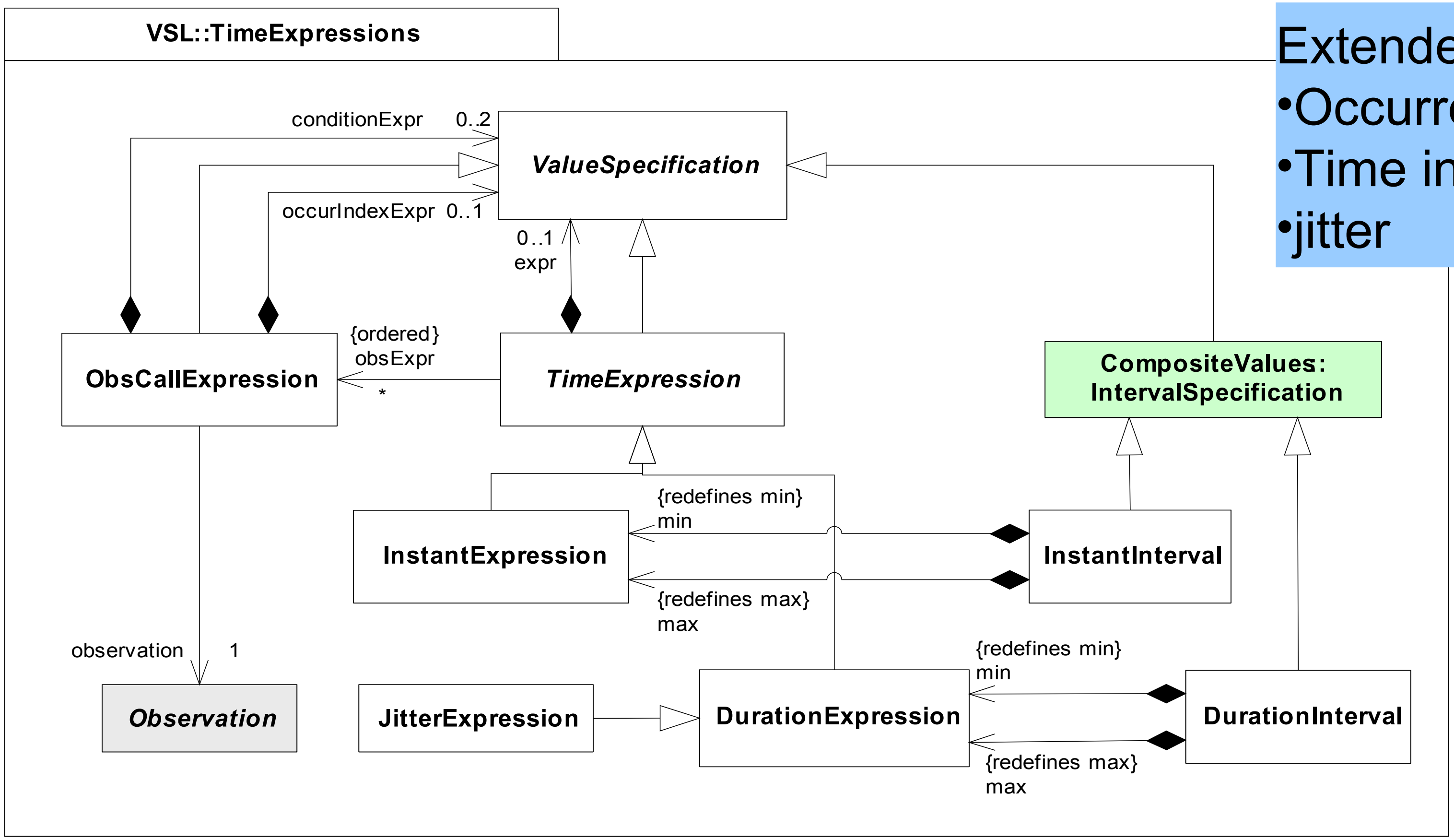


Domain

Instant
≠
Duration

Time specific languages:VSL Time Expressions

Expressing time values with EXPLICIT clocks



Extended capabilities:
- Occurrence index
- Time intervals
- jitter

Domain

Time Values: Concrete syntax

Examples of Clocked value expressions

- **Simple time values**

  (value=3.5, unit=ms, onClock='idealClk');

  3.5 ms on idealClk;

tuple, *a la* VSL

short form

- **Homogemeous expressions**

  (value=1.5, unit=ms, onClock='idealClk') +

  (value=150, unit=us, onClock='idealClk');

  → (value=1650, unit=us, onClock='idealClk')

Can be evaluated, because convFactor between units

- **Heterogeneous expressions**

  min (15 tick on prClk, 5 ms on idealClk);

Clock relation between prClk and idealClk must be provided

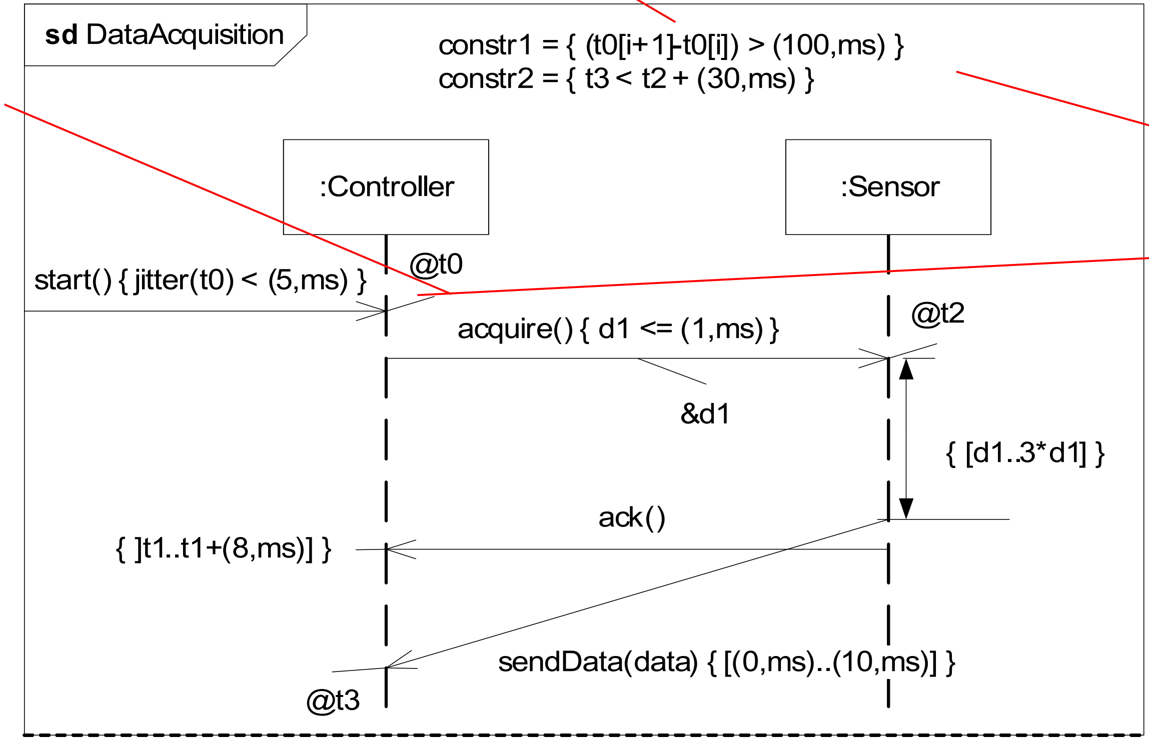
- **Additional capabilities with VSL**
  - Occurrence number, jitter,…
  - but implicitly on idealClk

Time specific languages:VSL Time Constraints

User

t0[i] denotes the i-th occurrence of

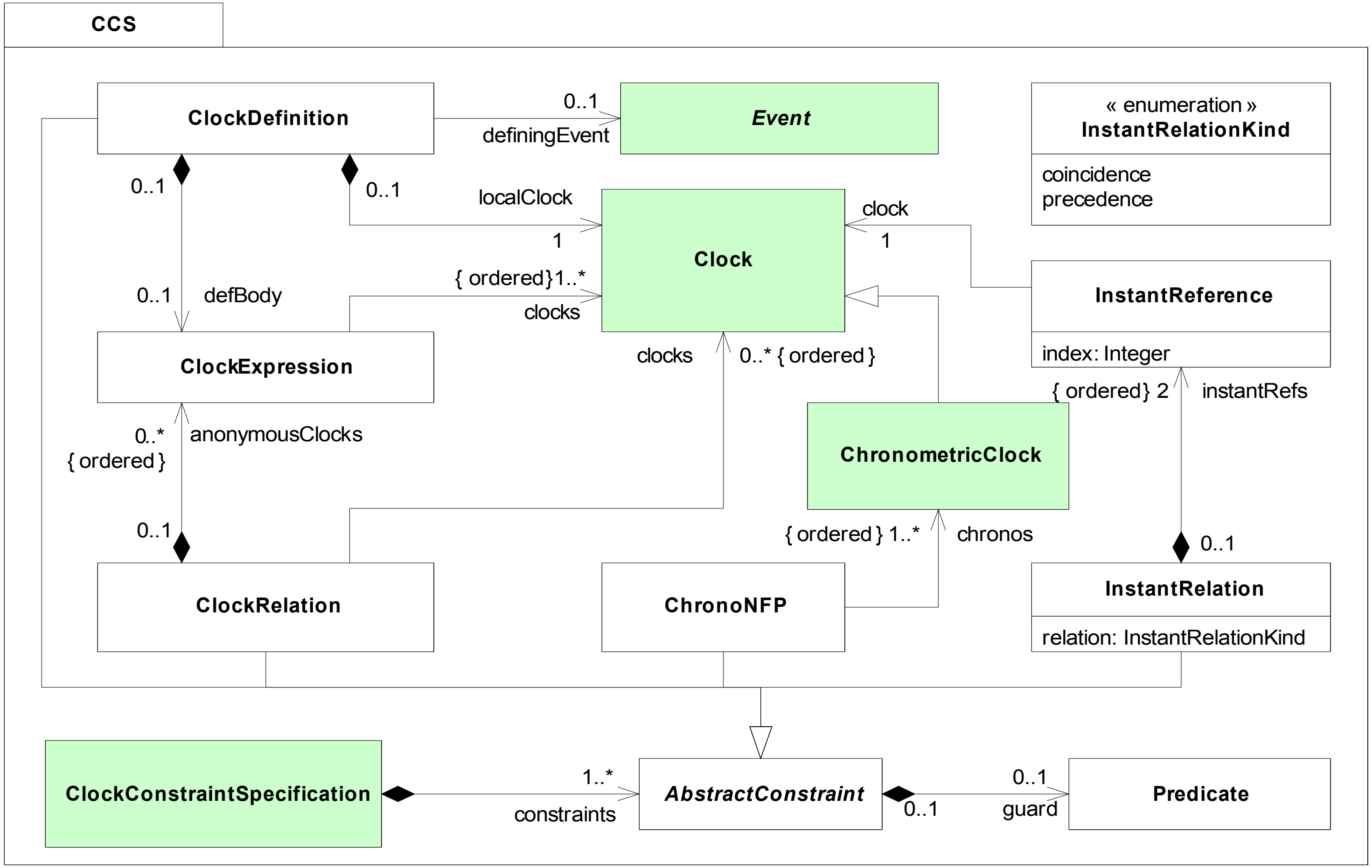
t0: observation of the message: start


sd DataAcquisition
constr1 = { (t0[i+1]-t0[i]) > (100,ms) }
constr2 = { t3 < t2 + (30,ms) }
:Controller
:Sensor
@t0
start() { jitter(t0) < (5,ms) }
@t2
acquire() { d1 <= (1,ms) }
&d1
{ [d1..3*d1] }
ack()
{]t1..t1+(8,ms)] }
sendData(data) { [(0,ms)..(10,ms)] }
@t3

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

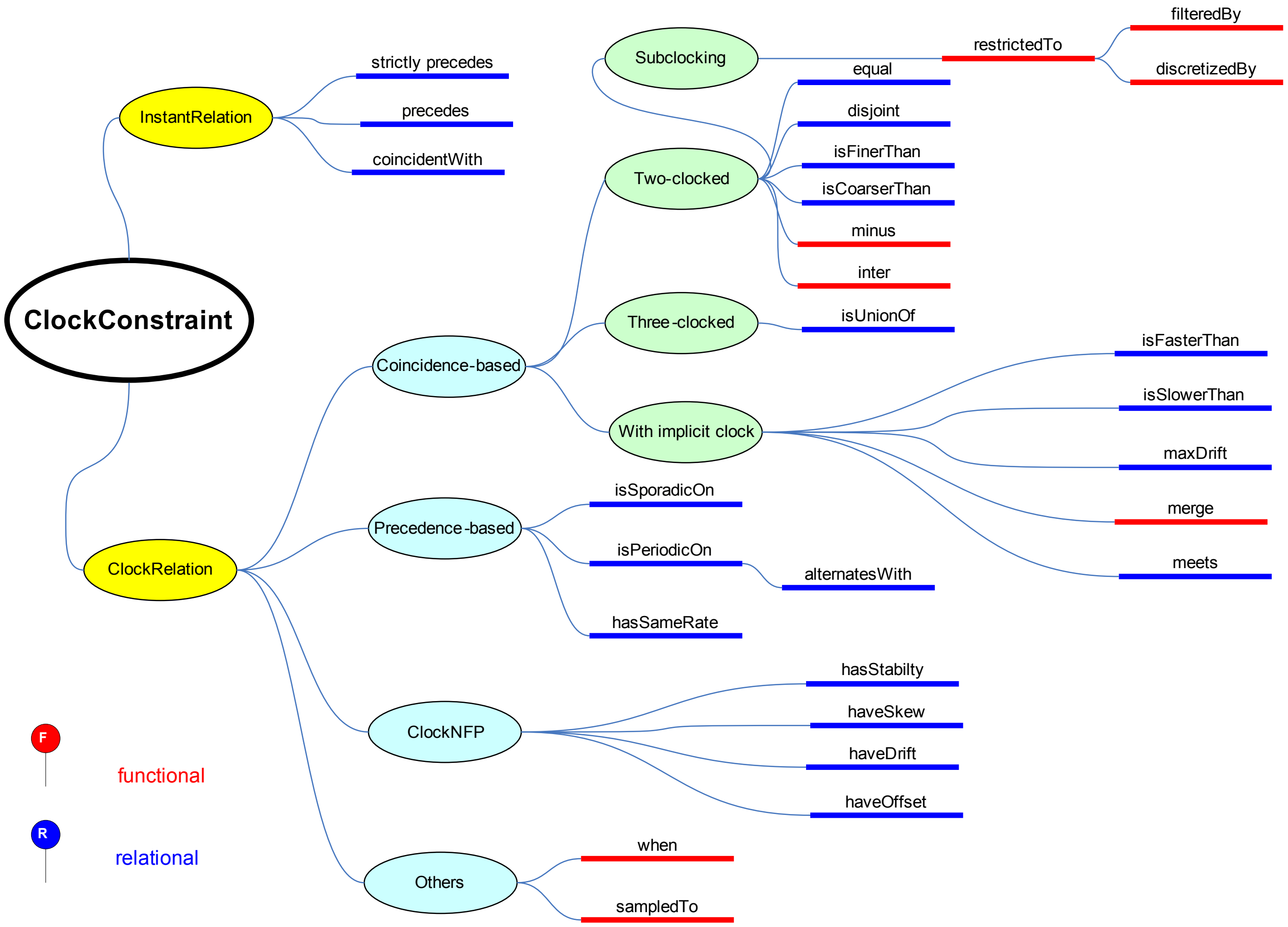
Time specific languages:Clock Constraint Specification

Domain



Expression of Clock dependencies

Clock Constraint Specification



Pre-defined Clock Constraints

Each relation has a mathematical specification

Overview

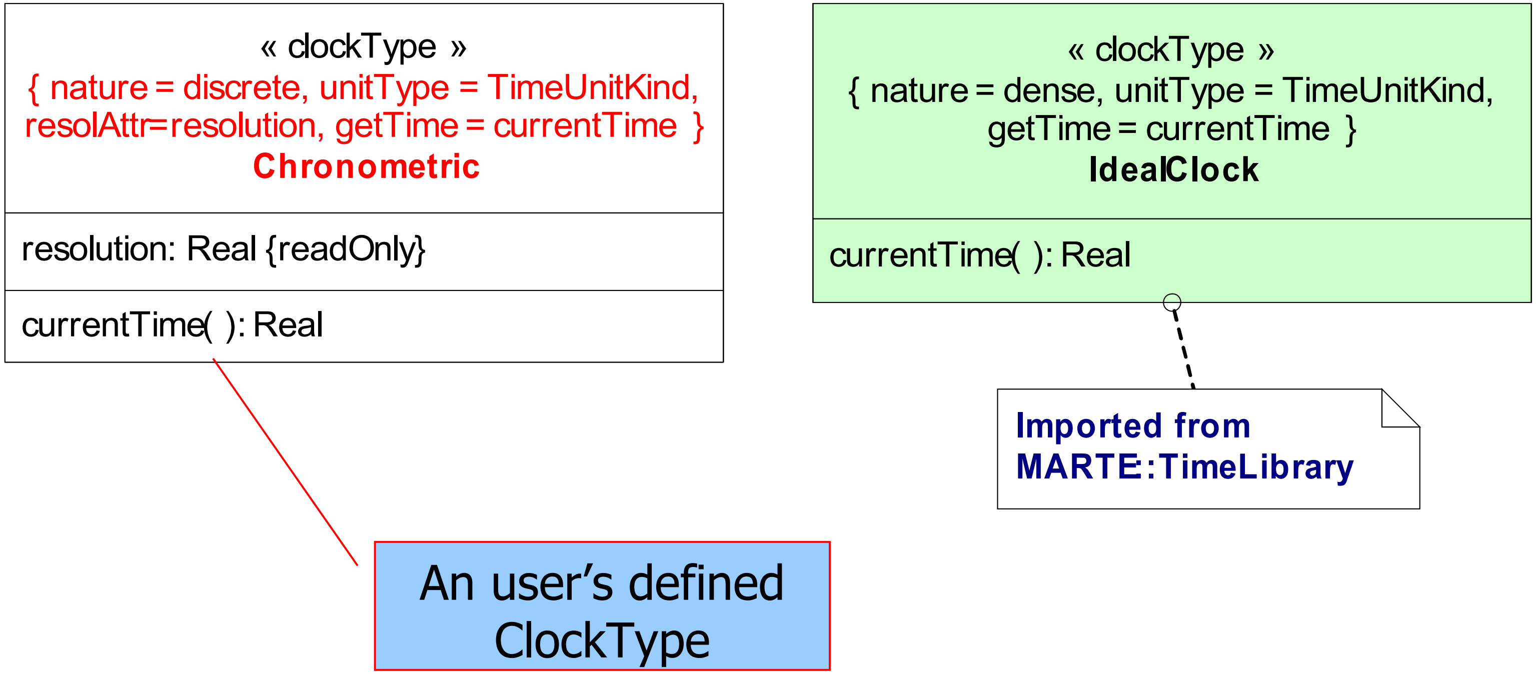
- SPT, UML 2 and Time
  - UML::CommonBehaviors::SimpleTime
- the MARTE Time domain view
  - a.k.a. the MARTE Time meta-model
  - Concepts and relationships
- the MARTE Time sub-profile
  - a.k.a. UML view
- **Usage of the Time sub-profile**

User

Chronometric Clocks (1)

How to specify chronometric clocks

User



« clockType »
{ nature = discrete, unitType = TimeUnitKind, resolAttr=resolution, getTime = currentTime }
Chronometric

resolution: Real {readOnly}

currentTime( ): Real

« clockType »
{ nature = dense, unitType = TimeUnitKind, getTime = currentTime }
IdealClock

currentTime( ): Real

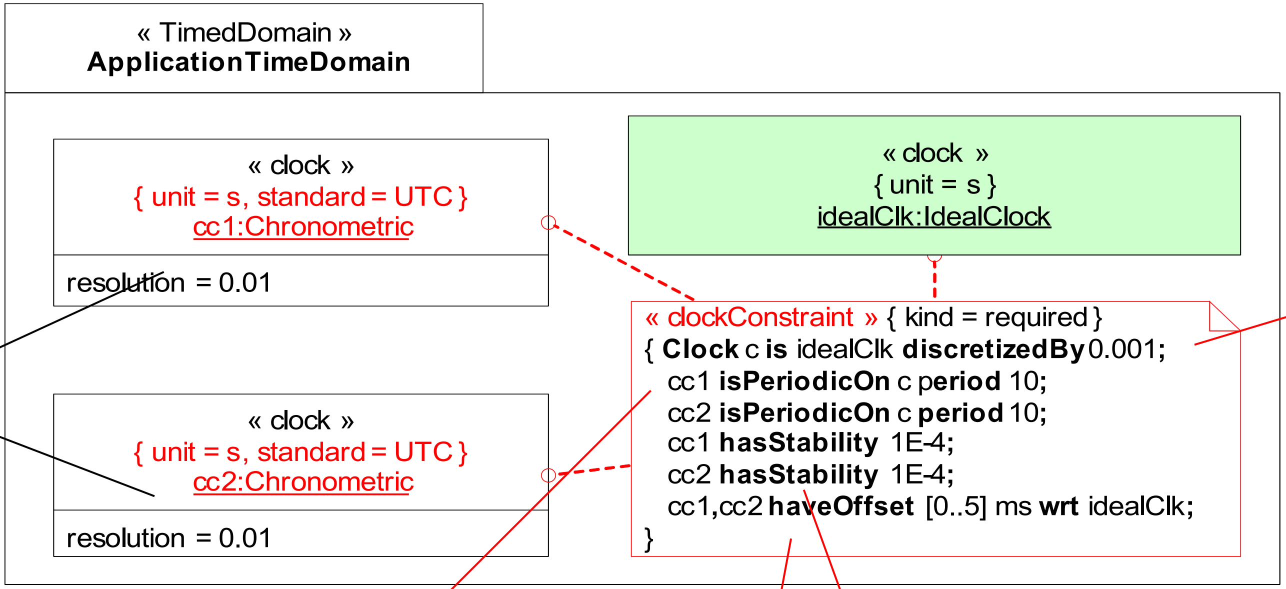
Imported from MARTE::TimeLibrary

An user's defined ClockType

Chronometric Clocks (2)

Specifying NFP of (non ideal) chronometric clocks

User



```
« clockConstraint » { kind = required }
{ 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;
}
```

Two instances

c: local ideal discrete clock – 1kHz

Exists d such that for all k:

$c[d+10*(k-1)] \prec cc1[k] \preccurlyeq c[d+10*k]$

$\Rightarrow 0<cc1[k+1]-cc1[k]<20$ **ms**

A Non-functional property:**Stability**

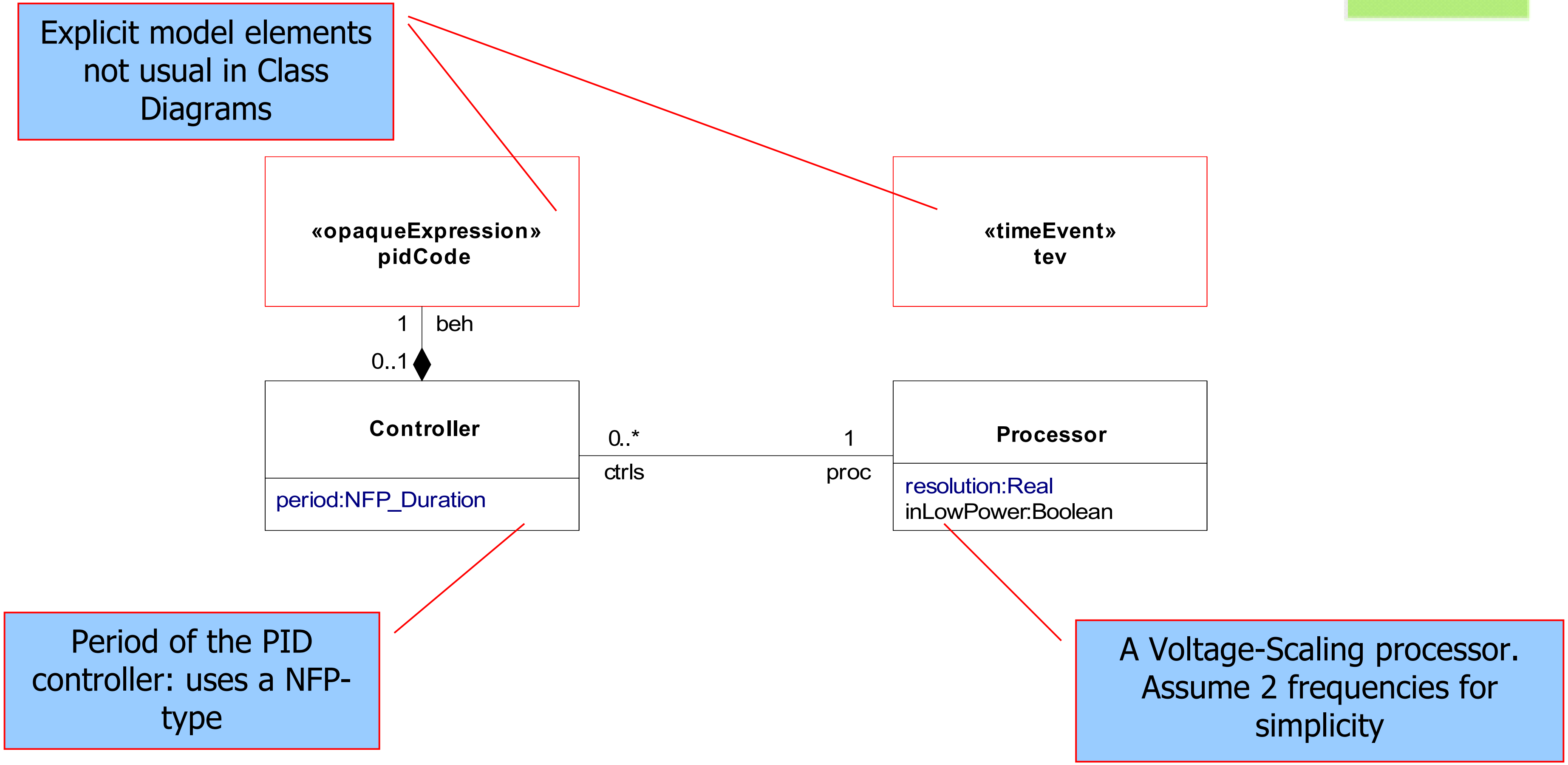
$10-0.001 \leq cc1[k+1]-cc1[k] \leq 10+0.001$ in **ms**

Another Non-functional property:**Offset**

Logical Clocks (1/4)

How to specify logical clocks:
1) Start with a standard UML class diagram

User



Explicit model elements not usual in Class Diagrams

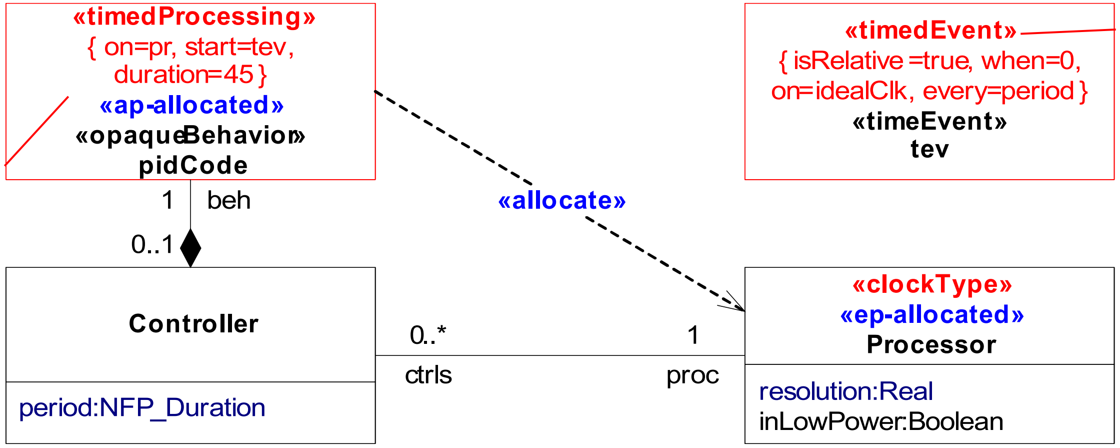
Period of the PID controller: uses a NFP-type

A Voltage-Scaling processor. Assume 2 frequencies for simplicity

Logical Clocks (2/4)

2) Apply MARTE stereotypes

User



The pid code is triggered by tev and takes 45 cycles of Processor

Event tev is periodic on idealClock, the period is the value of the controller's attribute

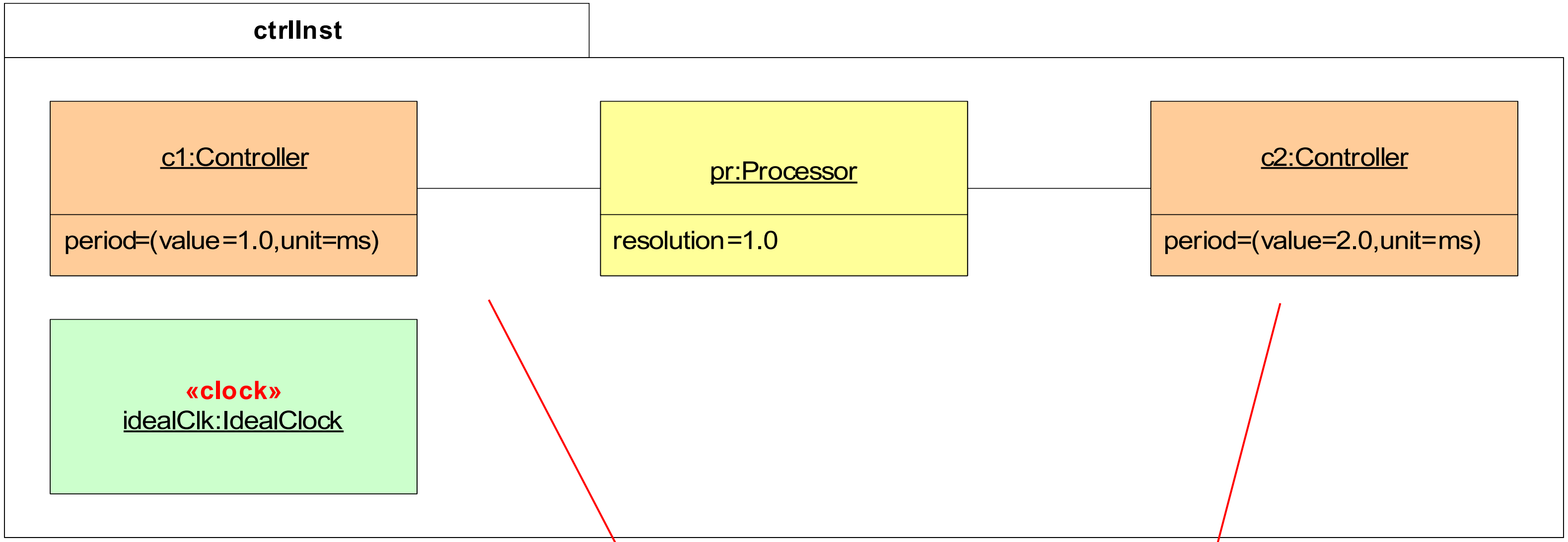
The class Processor is stereotyped by ClockType

Logical Clocks (3/4)

3) Instantiate user’s model elements

User

An instance of the system with an instance of Processor supporting two instances of Controller



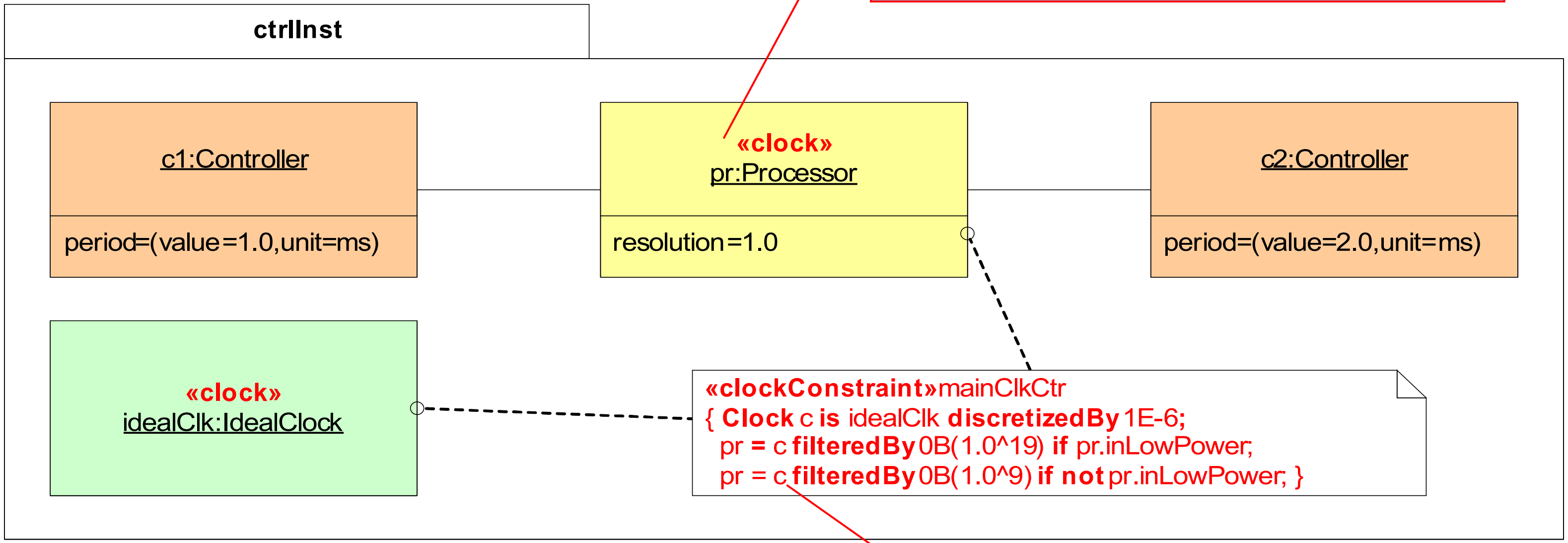
Each controller instance has its own period

Logical Clocks (4/4)

4) Introduce clock (by stereotyping)

User

This instance of Processor is used as a logical clock



```
«clockConstraint»mainClkCtr
{ Clock c is idealClk discretizedBy 1E-6;
  pr = c filteredBy 0B(1.0^19) if pr.inLowPower;
  pr = c filteredBy 0B(1.0^9) if not pr.inLowPower; }
```

This clock constraint binds processor clock cycle to physical time, taking account of the power mode

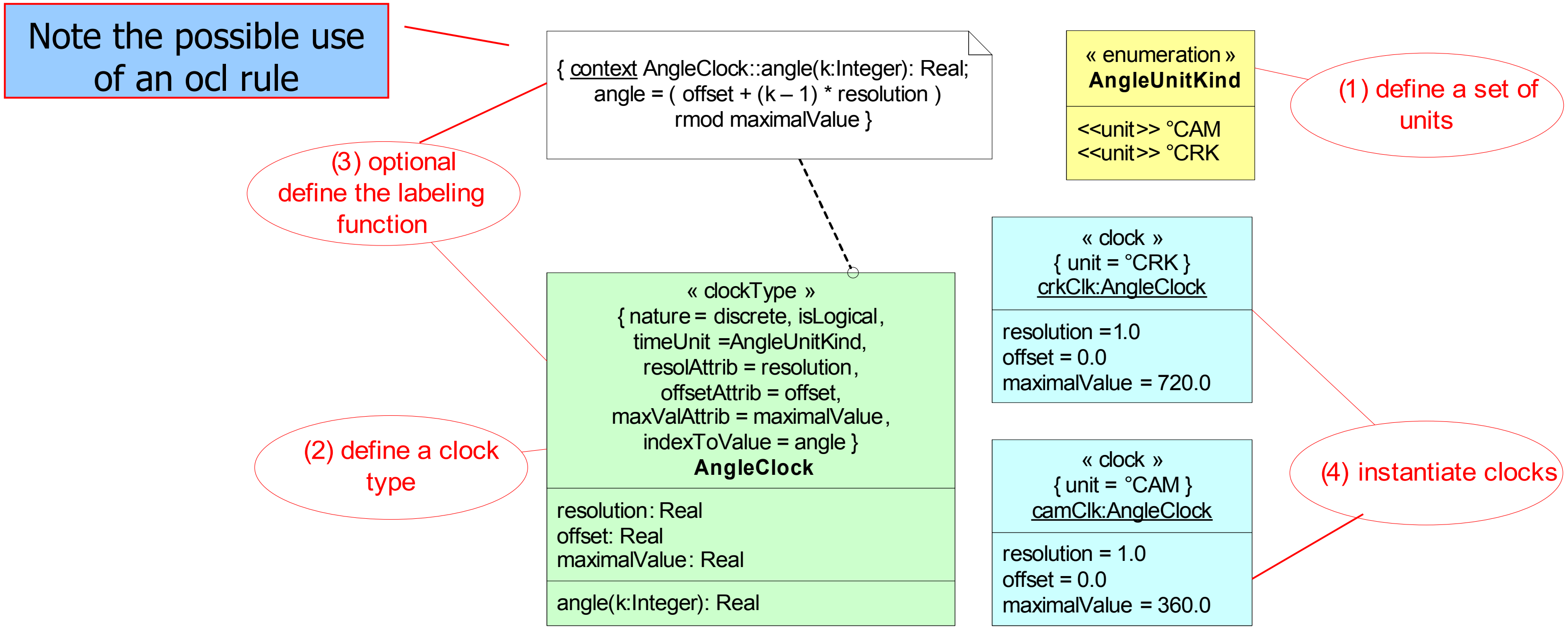
Multiform time (1/4)

Another example of logical clocks

Automotive application

For ignition and injection, the position of the camshaft or the crankshaft is a “natural” reference frame for events and behaviors.

=> Define logical clocks dealing with angular positions.



Note the possible use of an ocl rule

{ context AngleClock::angle(k:Integer): Real;
angle = (offset + (k – 1) * resolution)
rmod maximalValue }

« enumeration »
AngleUnitKind
<<unit>> °CAM
<<unit>> °CRK

(1) define a set of units

(3) optional define the labeling function

« clockType »
{ nature = discrete, isLogical,
timeUnit =AngleUnitKind,
resolAttrib = resolution,
offsetAttrib = offset,
maxValAttrib = maximalValue,
indexToValue = angle }
AngleClock
resolution: Real
offset: Real
maximalValue: Real
angle(k:Integer): Real

(2) define a clock type

« clock »
{ unit = °CRK }
crkClk:AngleClock
resolution =1.0
offset = 0.0
maximalValue = 720.0

« clock »
{ unit = °CAM }
camClk:AngleClock
resolution = 1.0
offset = 0.0
maximalValue = 360.0

(4) instantiate clocks

User

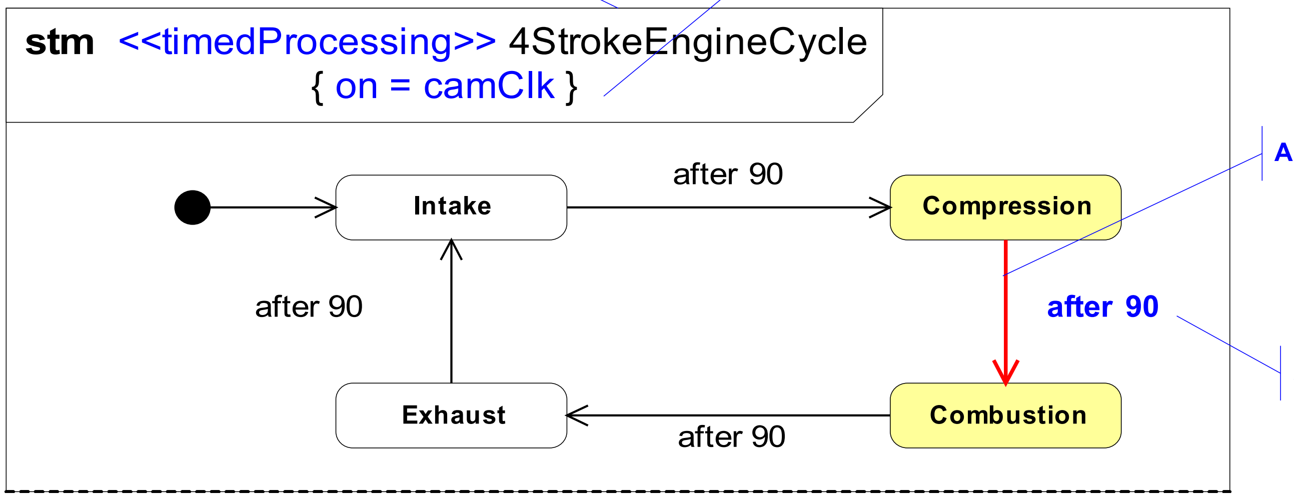
Multiform time (2/4)

Example of usage of an “AngleClock”

User

Stereotyped State Machine . Makes reference to a Clock

Reference to a (logical) clock, the unit of which is °CAM (elsewhere defined)



A transition

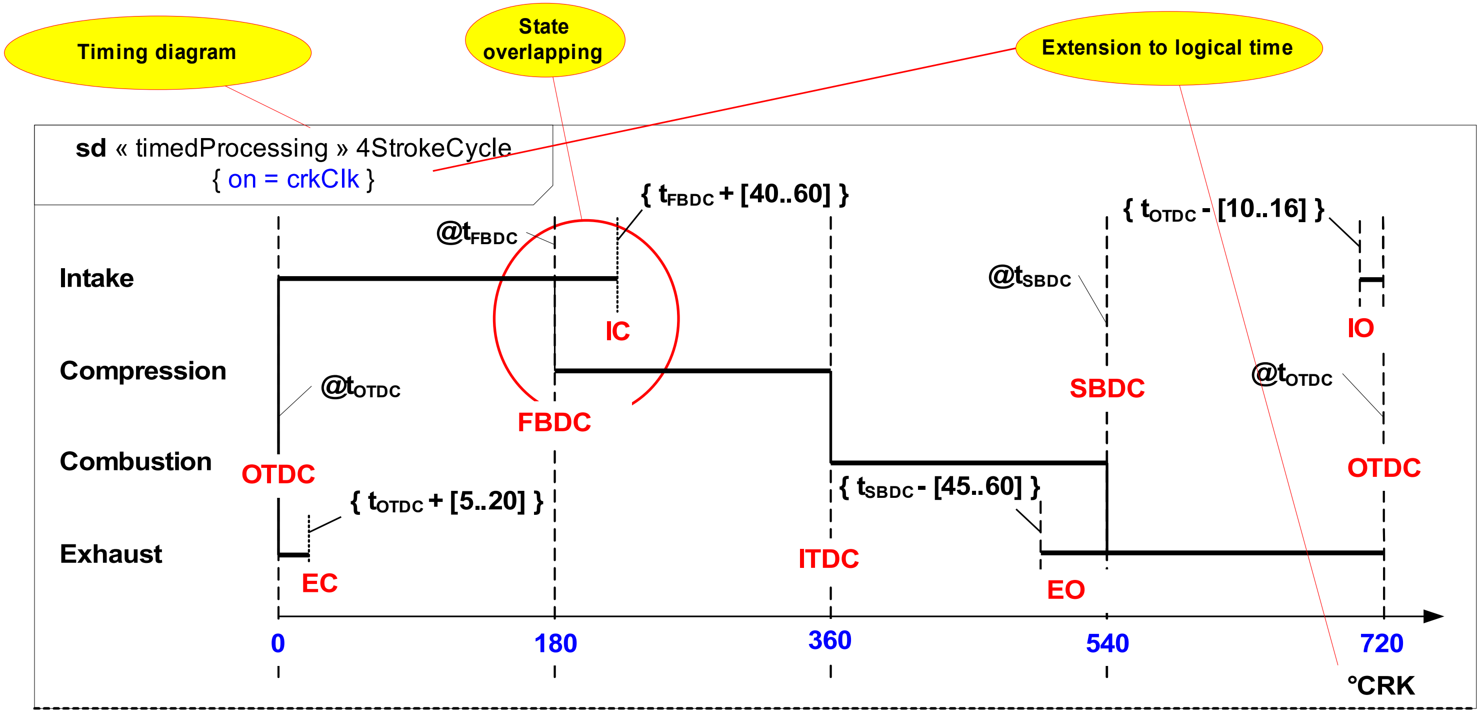
A trigger

Semantics:
90 °CAM after entering state *Compression* leave this state and enter state *Combustion*

Multiform time (3/4)

Another example of usage of an "AngleClock":
Enhanced timing diagram used in specification

User



crkClk: crankshaft Clock
°CRK: degree crank

TDC: Top Dead Center
ITDC: Ignition TDC
OTDC: Overlap TDC

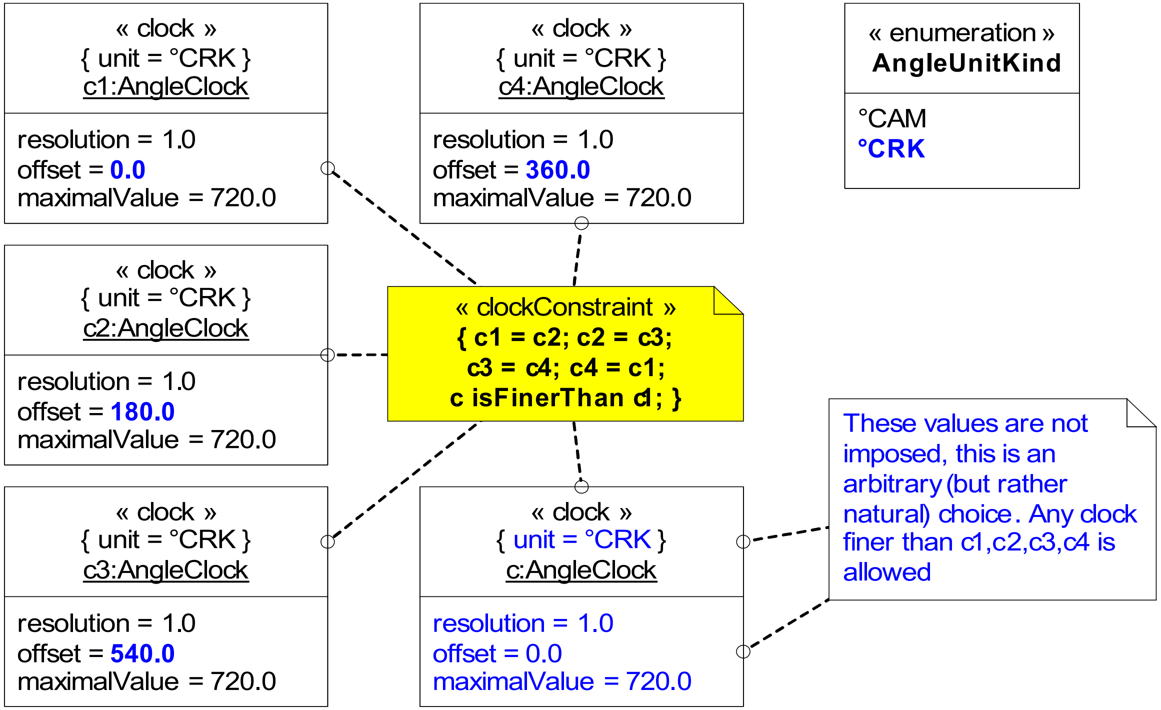
BDC: Bottom Dead Center
FBDC: First BDC
SBDC: Second BDC

IC: Intake closes
IO: Intake opens
EC: Exhaust closes
EO: Exhaust opens

Multiform time (4/4)

User

Combining logical clocks:
ck is an AngleClock used to specify the ignition of a cylinder
c is the clock used to specify ignitions in a 4-cylinder engine



These values are not imposed, this is an arbitrary (but rather natural) choice. Any clock finer than c1,c2,c3,c4 is allowed