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

Non-Functional Properties (NFPs)

Non-functional properties describe the “fitness” of systems behavior
(E.g., performance, memory usage, power consumption)

- **Nature of NFPs**
  - Quantitative: magnitude + unit (E.g., energy, data size, duration)
  - Qualitative (E.g., periodic or sporadic event arrival patterns)
- **NFP values need to be qualified**
  - E.g. source, statistical measure, precision,…
- **NFPs need to be parametric and derivable**
  - Variables: placeholders for unknown values
  - Expressions: math. and time expressions
- **NFPs need clear semantics**
  - Predefined NFPs (E.g., end-to-end latency, processor utilization)
  - User-specific NFPs (but still unambiguously interpreted!)

UML MARTE
www.omgmarte.org

Introduction to the MARTE's NFPs Framework

- UML lacks modeling capabilities for NFPs !!

- Value qualifiers?
- Measures?

UML Profile for NFPs

- NFP Libraries?
- Annotation mechanism?

- And UML expression syntax is also not sufficient!!

- Variables?
- **Structured Values?**

Value Specification Language (VSL)

- Data Type System?
- Complex time expressions?

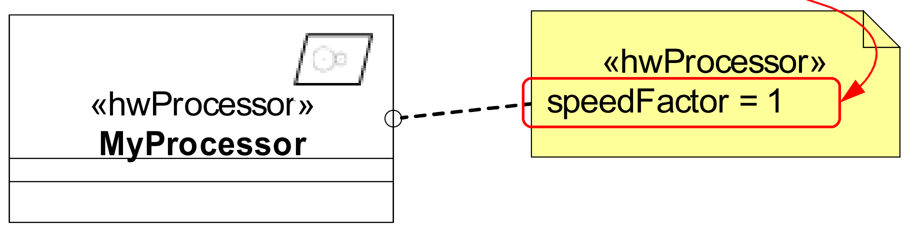
cea list THALES INRIA

The MARTE's NFP sub-profile

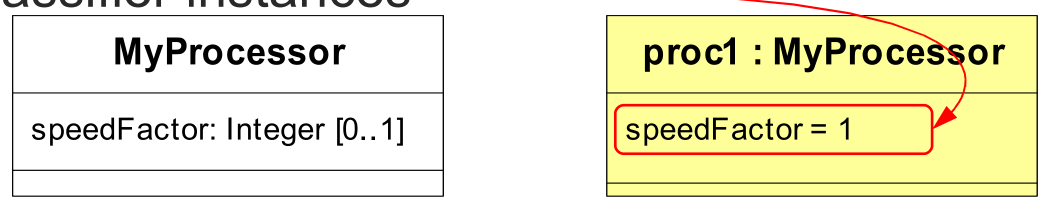
UML Profile for NFPs

Three mechanisms to annotate UML models:

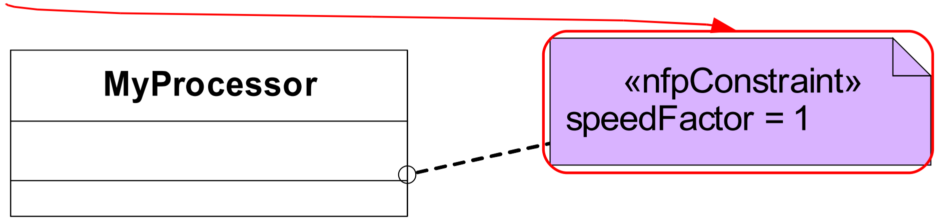
- Values of stereotype properties



- Slot values of classifier instances



- Constraints



Annotating NFPs in Tagged Values

MARTE pre-defined

1) Declare NFP types

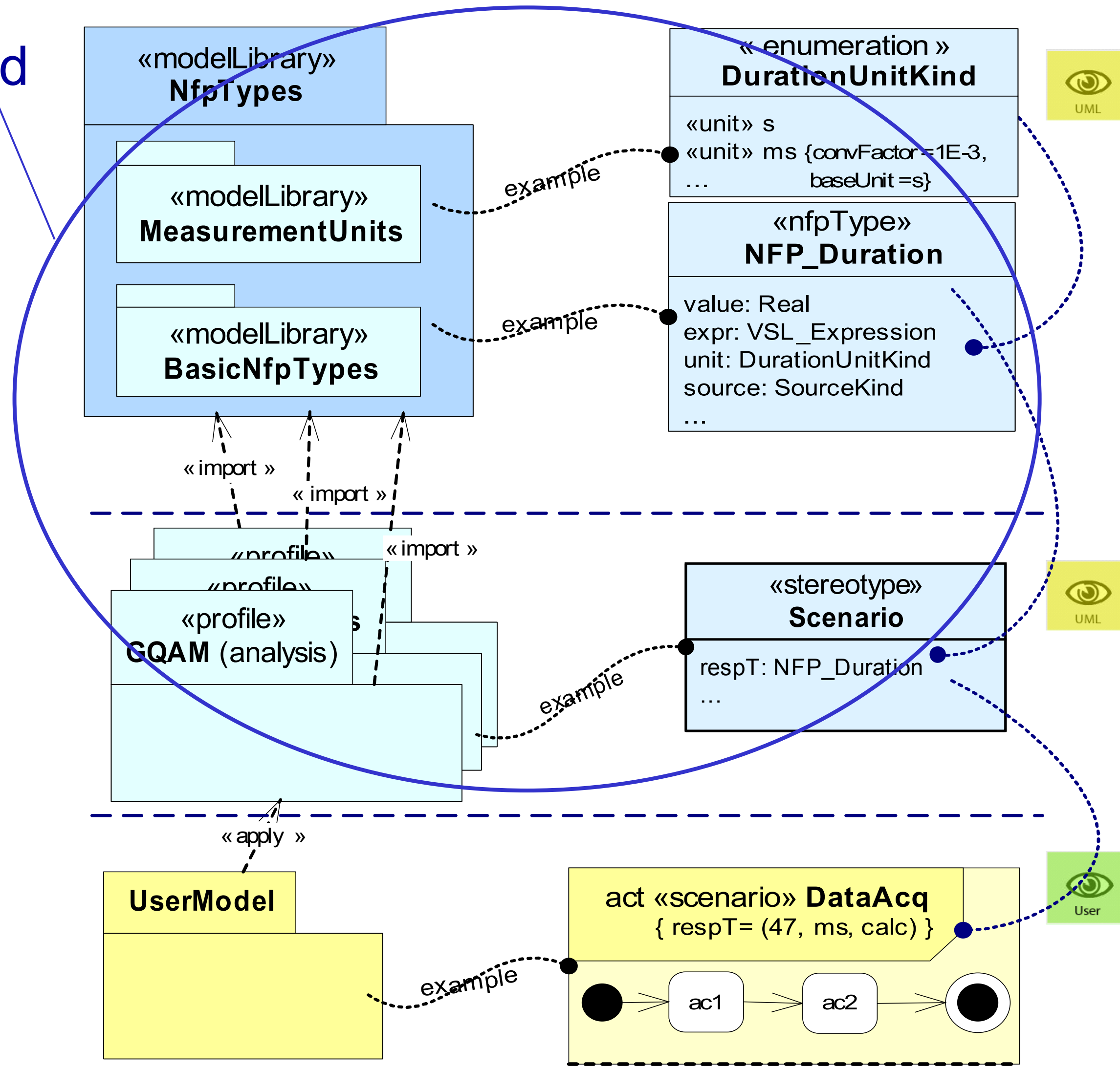
- Define measurement units and conversion parameters
- Define NFP types with qualifiers

2) Define NFP-like extensions

- Define stereotypes and their attributes using NFP types

3) Specify NFP values

- Apply stereotypes and specify their tag values using VSL



Annotating NFPs in Slots

1) Declare NFP types

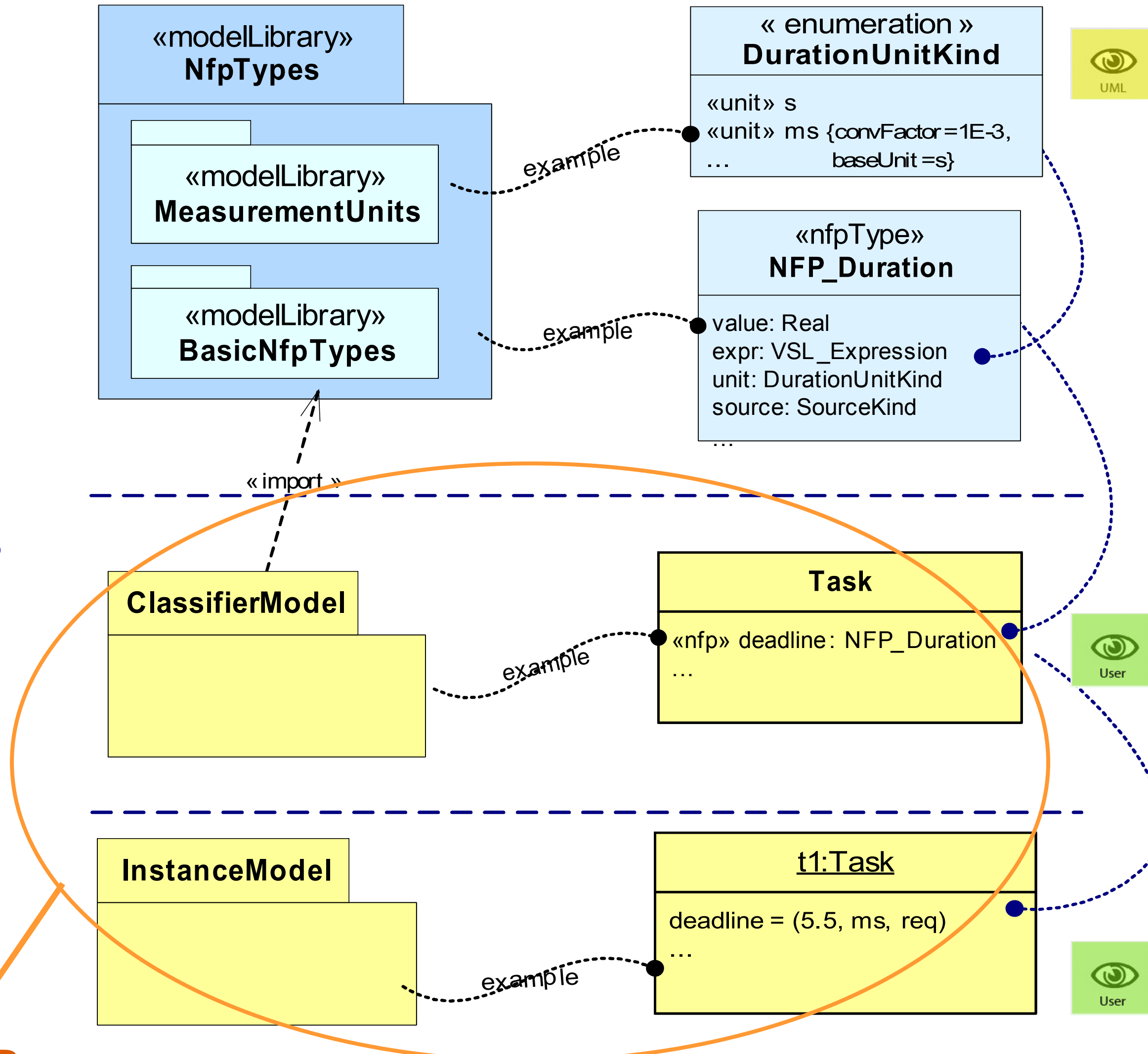
- Define measurement units and conversion parameters
- Define NFP types with qualifiers

2) Declare NFPs in user models

- Define classifiers and their attributes using NFP types
- Such attributes are tagged as «nfp»

3) Specify NFP values

- Instantiate classifiers and specify their slot values using VSL



Model-specific NFPs

Annotating NFPs in Constraints

1) Declare NFP types

- Define measurement units and conversion parameters
- Define NFP types with qualifiers

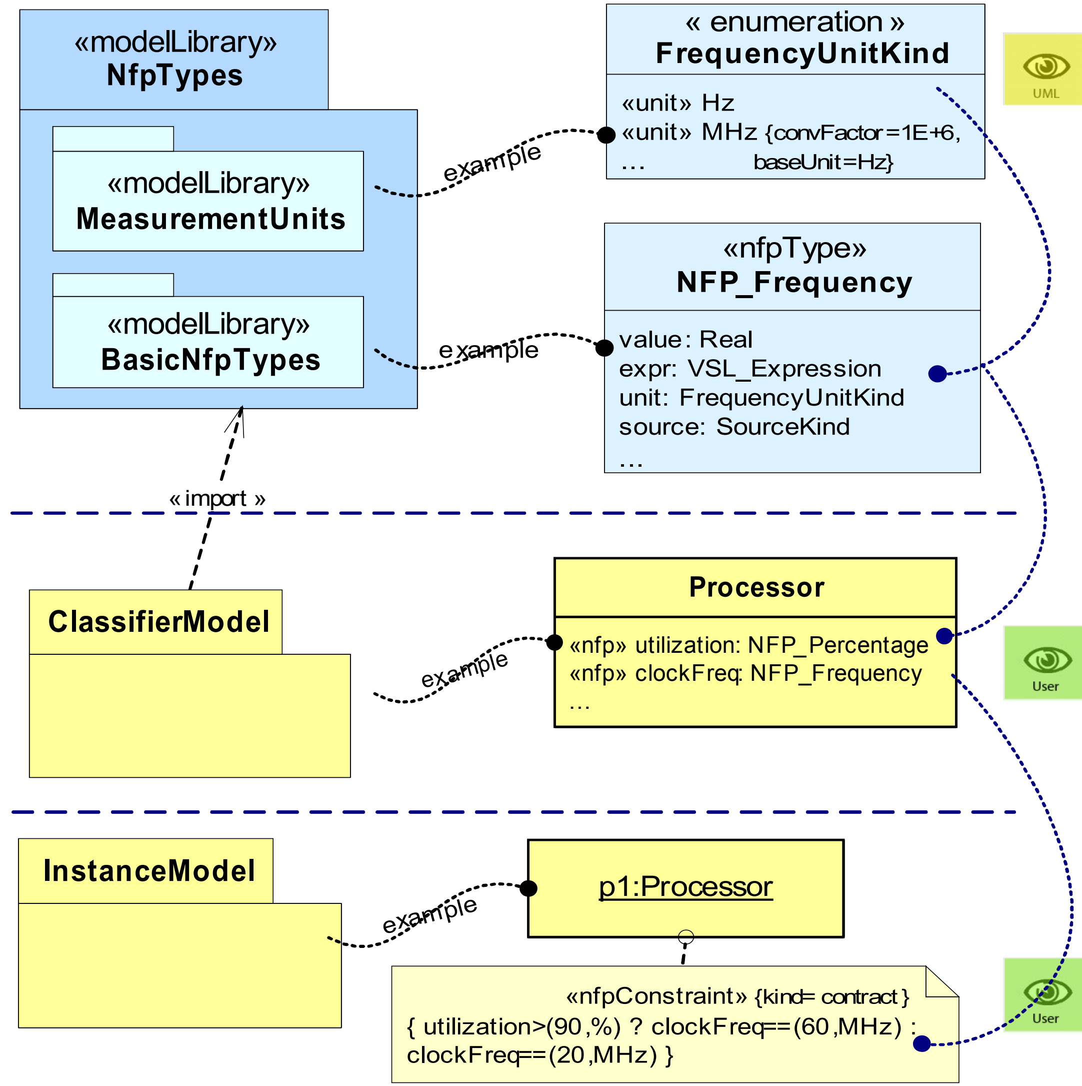
2) Declare NFPs

-Define classifiers and their attributes using NFP types

3) Specify NFP values

-Create Constraints to define assertions on NFP values using VSL

-«nfpConstraint» is a *required*, *offered*, or *contract* constraint of NFPs



The MARTE's NFP Modeling Framework

Value Specification Language (VSL)

- Three main language extensions to UML syntax
  - Grammar for extended expressions
  - Stereotypes for extended data types
  - Complex time expressions

Basic Textual Expressions in VSL

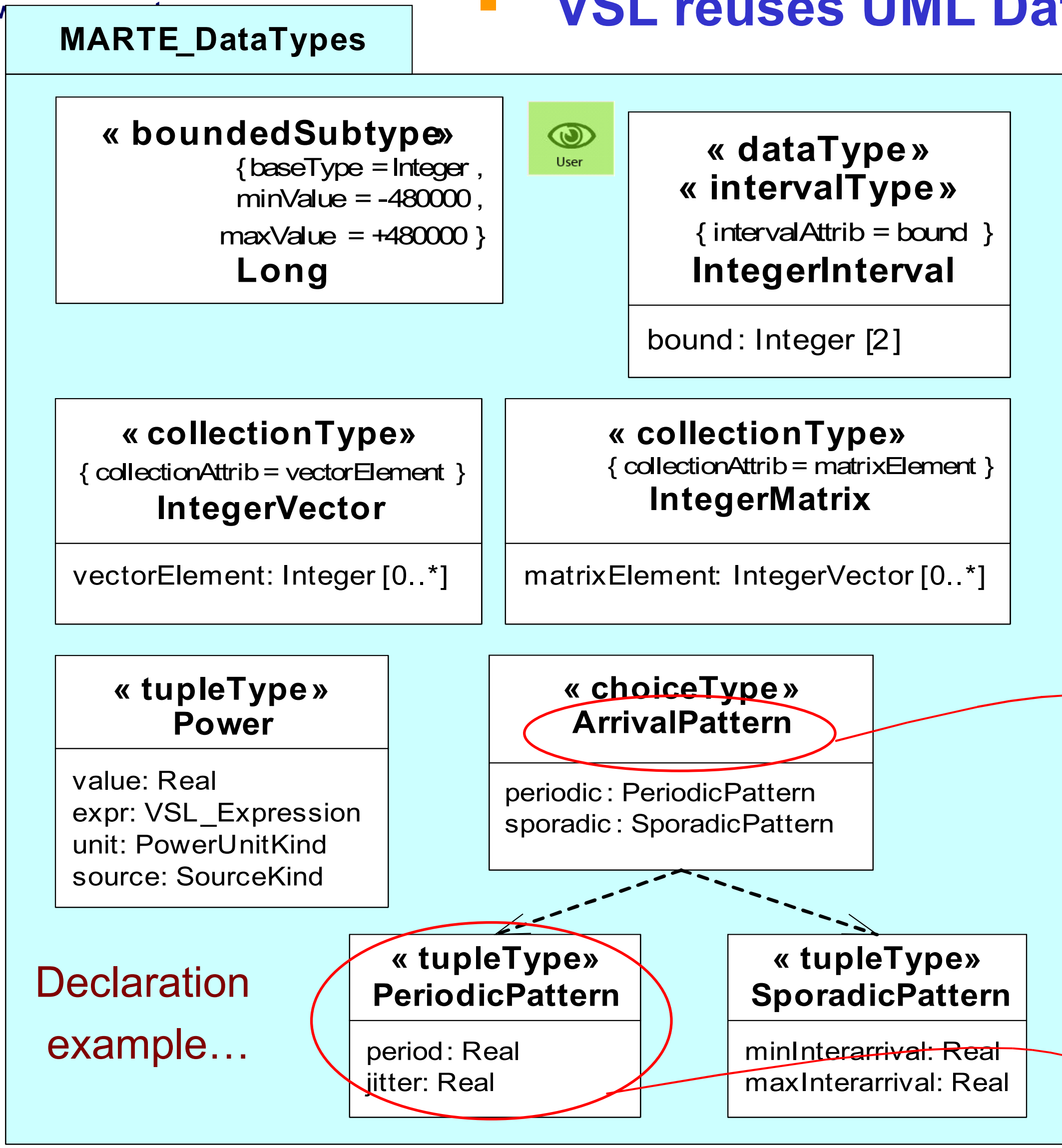
www.omgmarte.org

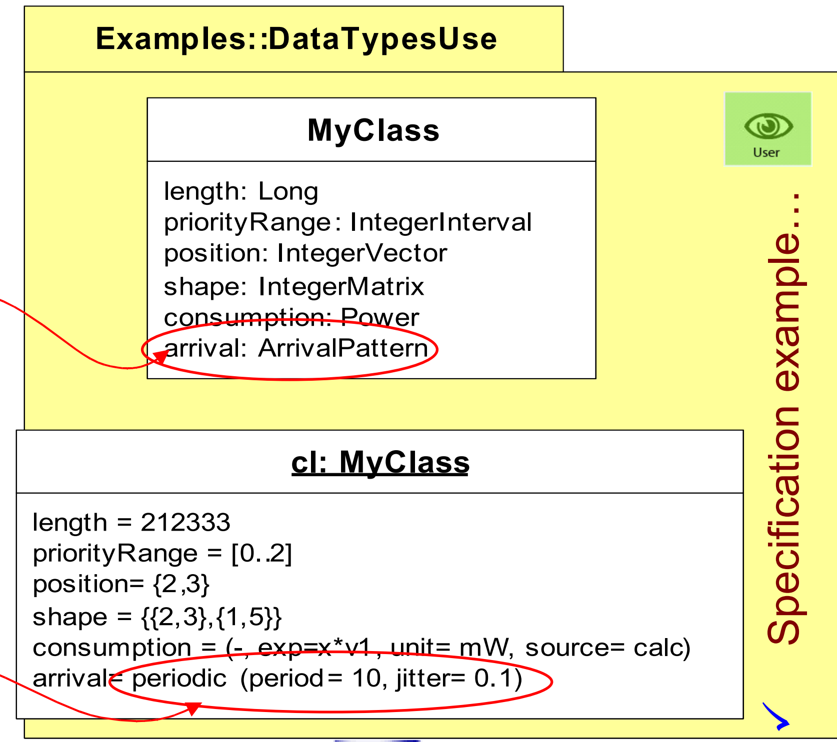
- Scope of the proposed extensions
  - Extended Primitive Values
  - Extended Composite Values
  - Extended Expressions

Value Spec.	Examples
Real Number	`1.2E-3          //scientific notation`
DateTime	`#12/01/06 12:00:00#   //calendar date time`
Collection	`{1, 2, 88, 5, 2}  //sequence, bag, ordered set..` `{{1,2,3}, {3,2}} //collection of collections`
Tuple and choice	`(value=2.0, unit= ms)     //duration tuple value` `periodic(period=2.0, jitter=3.3) //arrival pattern`
Interval	`[1..251[  //upper opened interval between integers` `[$A1..$A2]      //interval between variables`
Variable declaration & Call	`io$var1      //input/output variable declaration` `var1      //variable call expression.`
Arithmetic Operation Call	`+(5.0,var1) //"add" operation on Real datatypes` `5.0+var1    //infix operator notation`
Conditional Expression	`((var1<6.0)?(10^6):1)  //if true return 10 exp 6,else 1`

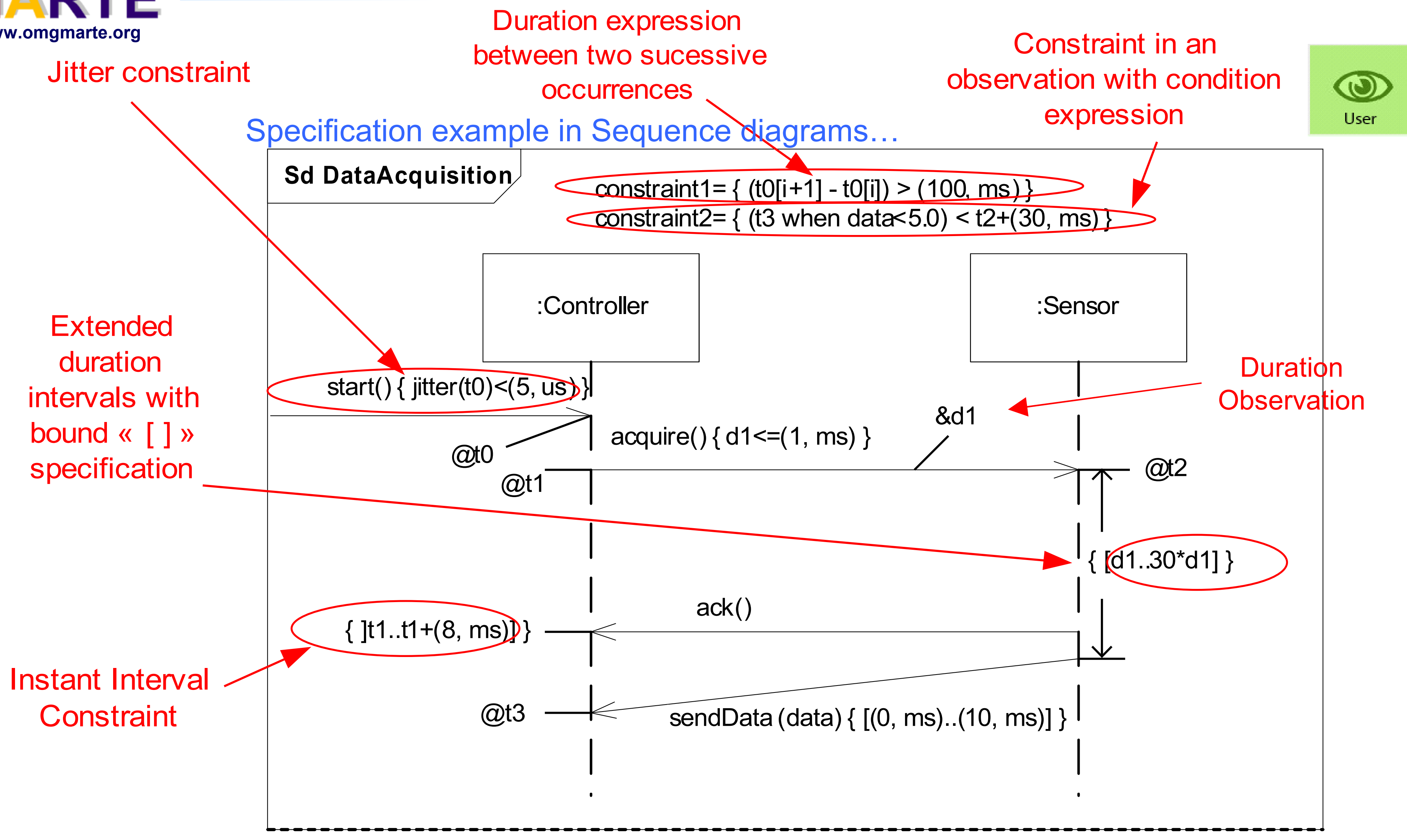
VSL Extended Data Types

- VSL reuses UML DataType constructs, but adds…
  - BoundedSubtype
  - IntervalType
  - CollectionType
  - TupleType
  - ChoiceType





Examples of Time Expressions with VSL



Conclusions on MARTE::NFPs

- **Synthesis of best modeling practices…**
  - OCL: full constraint language, but hard to use and not real-time oriented
  - SPT Profile: built-in TVL language is simpler, but not flexible
  - QoS&FT Profile: annotation mechanism is flexible, but complex
  - ➔ NFP & VSL reuse selected modeling features, while still providing simplicity and flexibility

- **Foundations…**
  - Reuse OCL constructs: grammar for values and expressions
  - Generic data type system: (based on ISO's General-Purpose Datatypes)
  - VSL extends UML Simple Time model (e.g. occurrence index, jitters)
  - Formally defined by abstract and concrete syntaxes (grammar)