www.omgmarte.org

MARTE Tutorial

An OMG standard:
UML profile to develop Real-Time and Embedded systems
Overview, NFP / VSL, GCM

Context of This Work

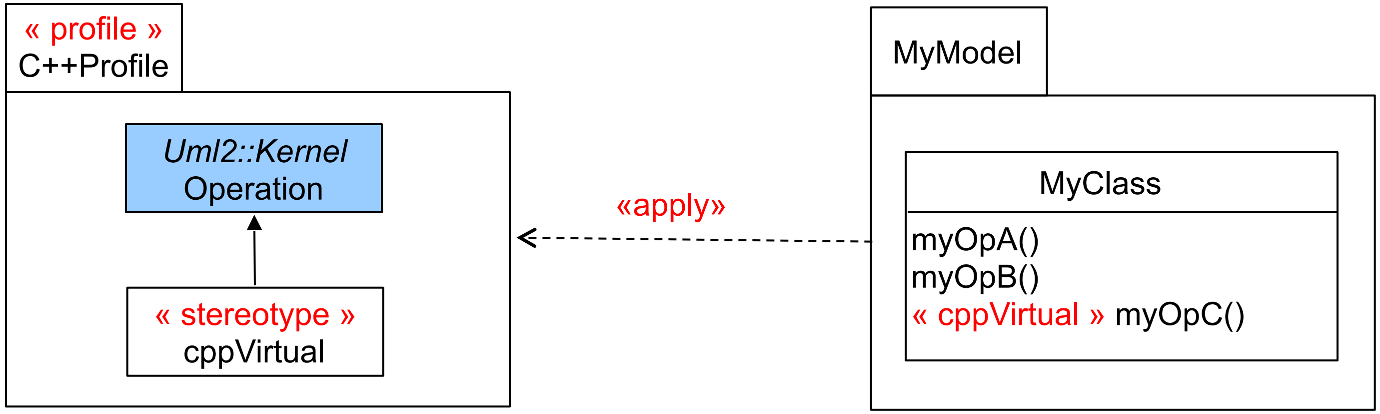
- The present courseware has been elaborated in the context of the MODELPLEX European IST FP6 project (http://www.modelplex.org/).
- Co-funded by the European Commission, the MODELPLEX project involves 21 partners from 8 different countries.
- MODELPLEX aims at defining and developing a coherent infrastructure specifically for the application of MDE to the development and subsequent management of complex systems within a variety of industrial domains.
- To achieve the goal of large-scale adoption of MDE, MODELPLEX promotes the idea of a collaborative development of courseware dedicated to this domain.
- The MDE courseware provided here with the status of open-source software is produced under the EPL 1.0 license.

Agenda

- **Part 1**
  - **Marte Overview**
- **Part 2**
  - A component model for RT/E
- **Part 3**
  - Non-functional properties modeling

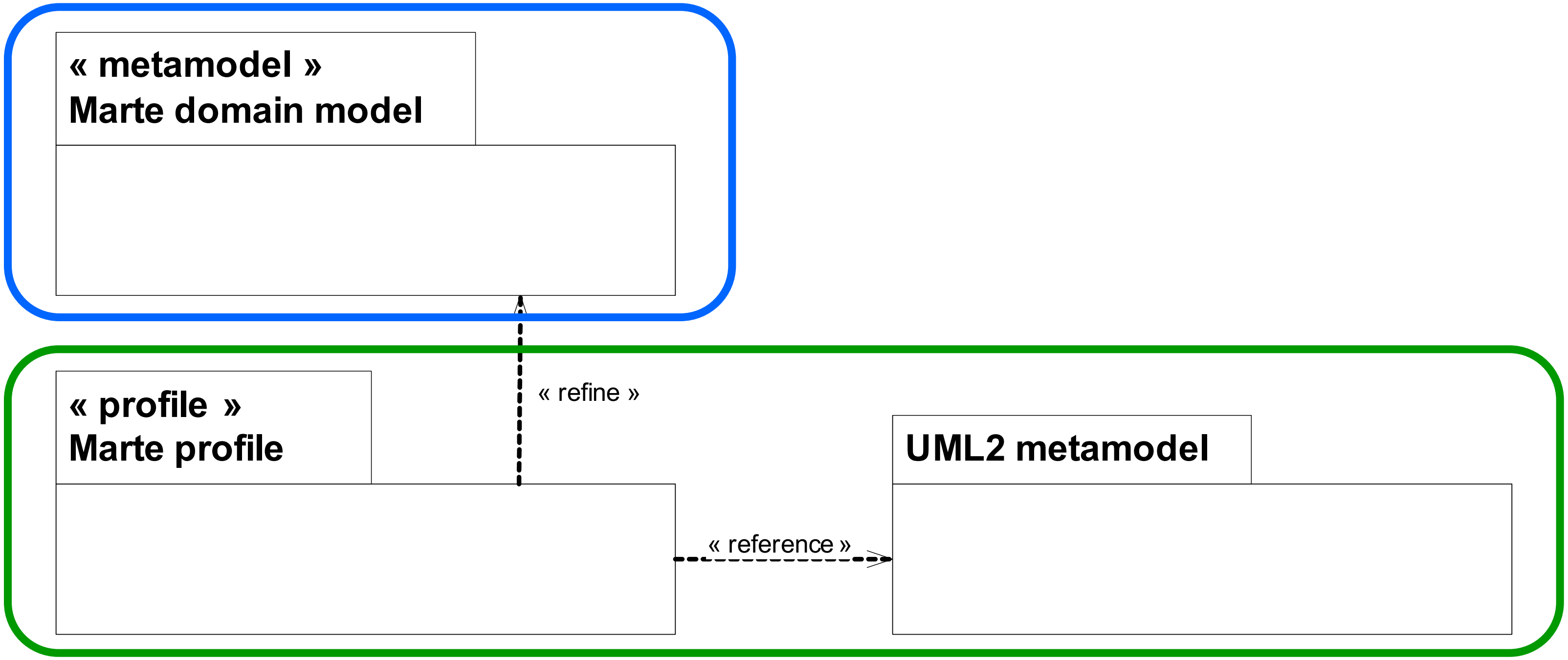
UML Profiling Basics

- **Profile:**
  - Lightweight Extension / Specialization of the UML metamodel
    - For particular application domains
  - Stereotypes
    - Extension / Specialization of existing metaclasses
    - Contains a set of “tagged values” (i.e. domain specific properties)
  - Constraints
    - On the usage of stereotypes
    - On the usage of constructs provided by the source metamodel
  - Notation options (i.e. icons, figures)



Design Pattern Adopted for the MARTE Profile

- **Stage 1 → Description of MARTE** domain models (DV)
  - Purpose: Formal description of the concepts required for MARTE
  - Techniques: Meta-modeling
- **Stage 2 → Mapping of MARTE domain models towards UML2:** UML Representation (UR)
  - Purpose: MARTE domain models design as a UML2 extensions
  - Techniques: UML2 profile



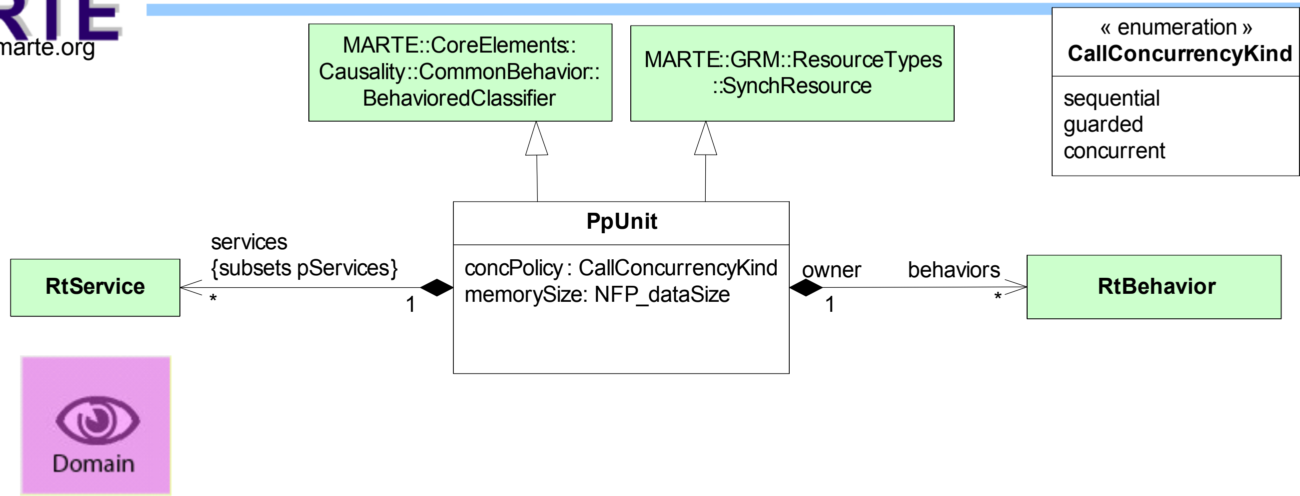
How to read this tutorial

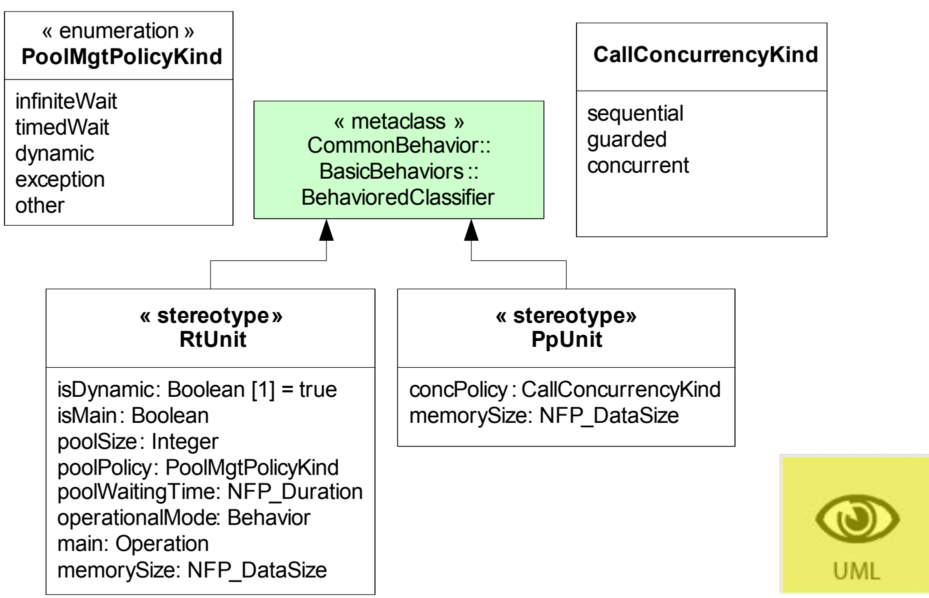
- Within next slides, we may shown models at different levels of abstraction. We will clarify each level through following pictograms
  - For Domain View level

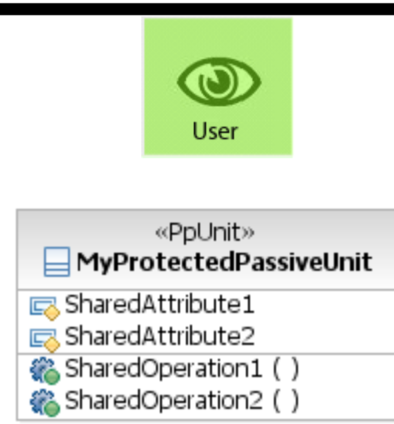
  - For UML Profile View Level

  - For User Model View Level

Example: Domain model → Profile → Usage







Relationships with other OMG Standards

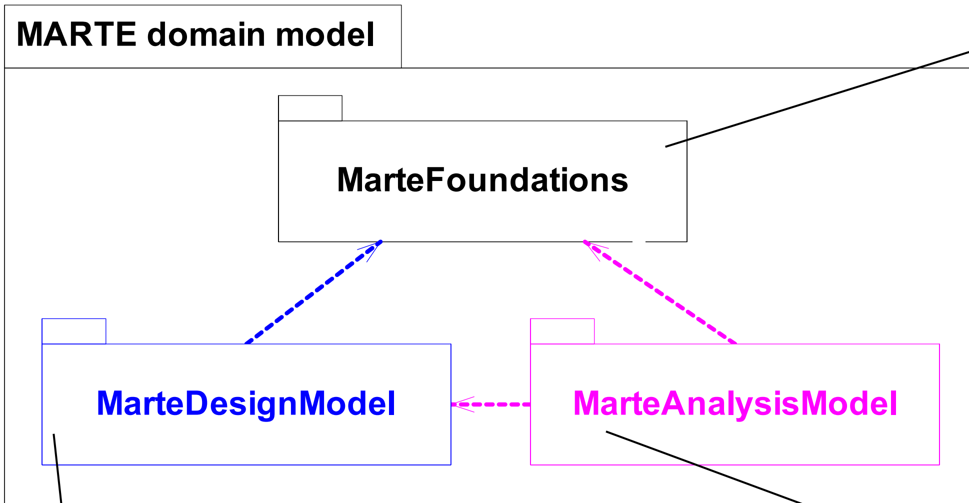
www.omgmarte.org

- **Relationships with generic OMG standards**
  - Profile the UML2 superstructure meta-model
  - Replace UML Profile for SPT (Scheduling, Performance and Time)
  - Use OCL2 (Object Constraints Language)

- **Relationships with RT&E specific OMG standards**
  - Existing standards
    - The UML profile for Modeling QoS and FT Characteristics and Mechanisms
      - **Addressed through MARTE NFP package (in a way detailed in the NFP presentation)**
    - The UML profile for SoC (System On Chip)
      - **More specific than MARTE purpose**
    - The Real-Time CORBA profile
      - **Real-Time CORBA based architecture can be annotated for analysis with Marte**
    - The UML profile for Systems Engineering (SysML)
      - **Specialization of SysML allocation concepts and reuse of flow-related concepts**
      - **Ongoing discussion to include VSL in next SysML version**
      - **Overlap of team members**

MARTE Overview

Domain



Foundations for RT/E systems modeling and analysis:
- ➔ CoreElements
- ➔ NFPs
- ➔ Time
- ➔ Generic resource modeling
- ➔ Generic component modeling
- ➔ Allocation

Specialization of MARTE foundations for modeling purpose (specification, design, …):
- ➔ RTE model of computation and communication
- ➔ Software resource modeling
- ➔ Hardware resource modeling

Specialization of foundations for annotating model for analysis purpose:
- ➔ Generic quantitative analysis
- ➔ Schedulability analysis
- ➔ Performance analysis

Extracted from S.Gerard (ECRTS07)

MARTE Frontiers and Challenges

- **MARTE define the language constructs only!**
  - Common patterns, base building blocks, standard NFP annotations
  - Generic constraints that do not force specific execution models, analysis techniques or implementation technologies
- **It does not cover methodologies aspects:**
  - Interface-Based Design, Design Space Exploration
  - Means to manage refinement of NFP measurement models
  - Concrete processes to storage, bind, and display NFP context models
  - Mapping to transform MoCCs into analysis models

MARTE is to the RTES domain as UML to the System & Software domain: a family of large and open specification formalisms!

Agenda

- **Part 1**
  - Marte Overview
- **Part 2**
  - **A component model for RT/E**
- **Part 3**
  - Non-functional properties modeling

Component-based paradigms in the RTE domain

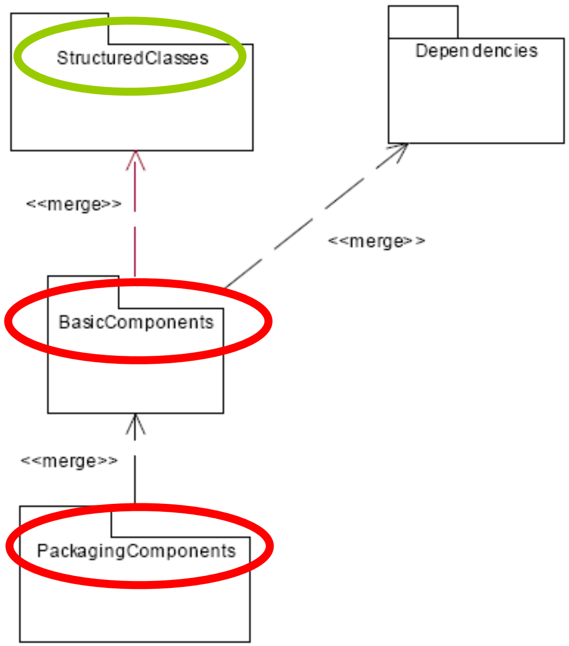
- **Component architectures are increasingly used in RTE execution platforms**
  - Need for manageable and reusable pieces of software
  - Key examples: Lightweight-CCM, SCA, Autosar

- **Concept of component also used to structure System / Software engineering processes**
  - Entities under analysis/design broken down into a series of components
  - Applicable at different stages of the process
  - Different kind: active vs. passive (e.g., UML active classes)
  - Examples of related languages: SysML, AADL

There is a need to provide modeling constructs to support these concepts at different levels of abstraction

What is a component in UML?

- **UML distinguishes the notions of structured class and component**
  - The kernel of the language defines *Class* and *Interface*
  - *StructuredClasses* defines *Port* and *Connector* and provide the ability to describe a *Class* as an assembly of parts
  - *Basic* and *PackagingComponent* define the notion of component realization and adds packaging capabilities

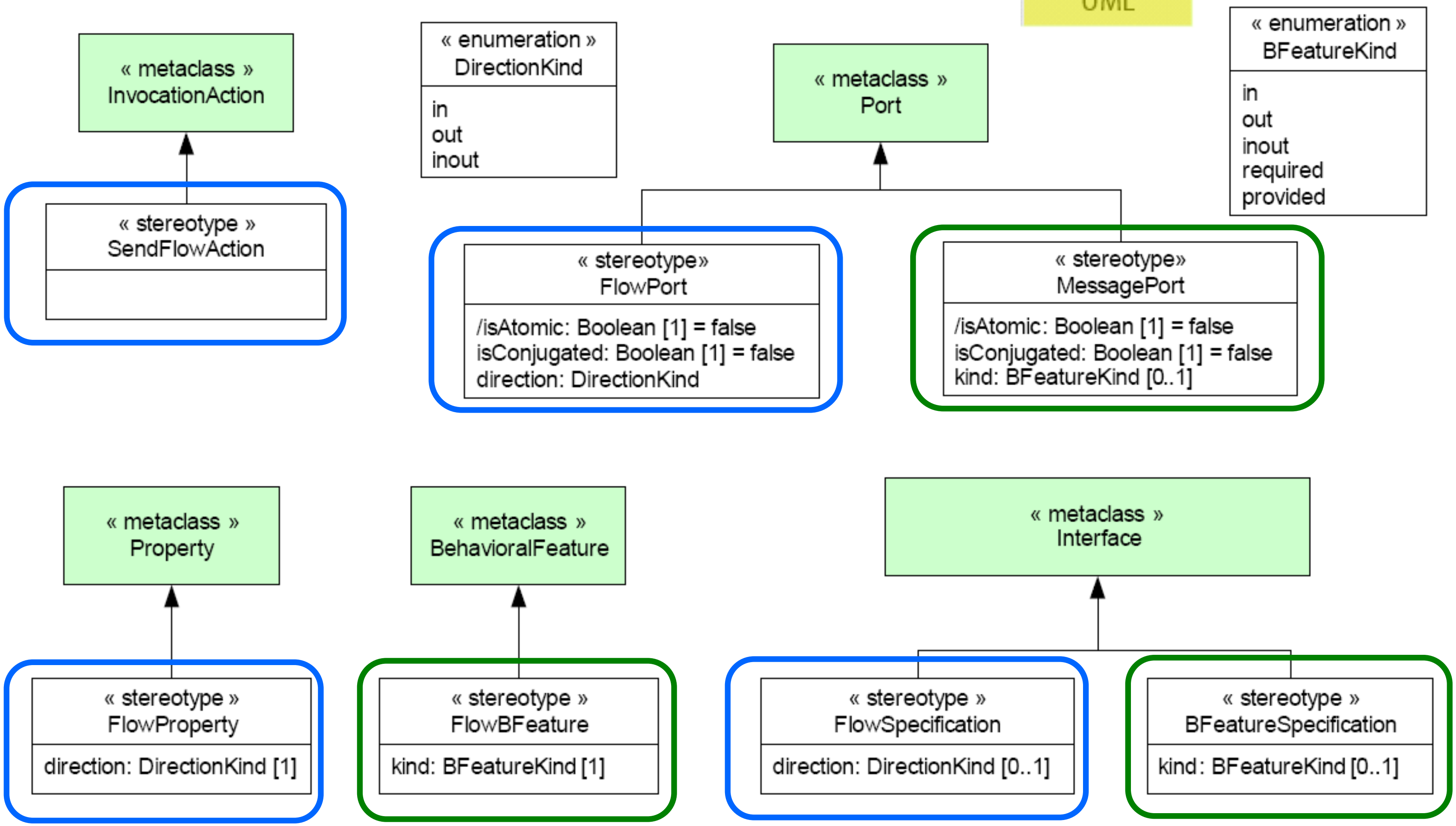


- In any case, no support for flow-oriented communications

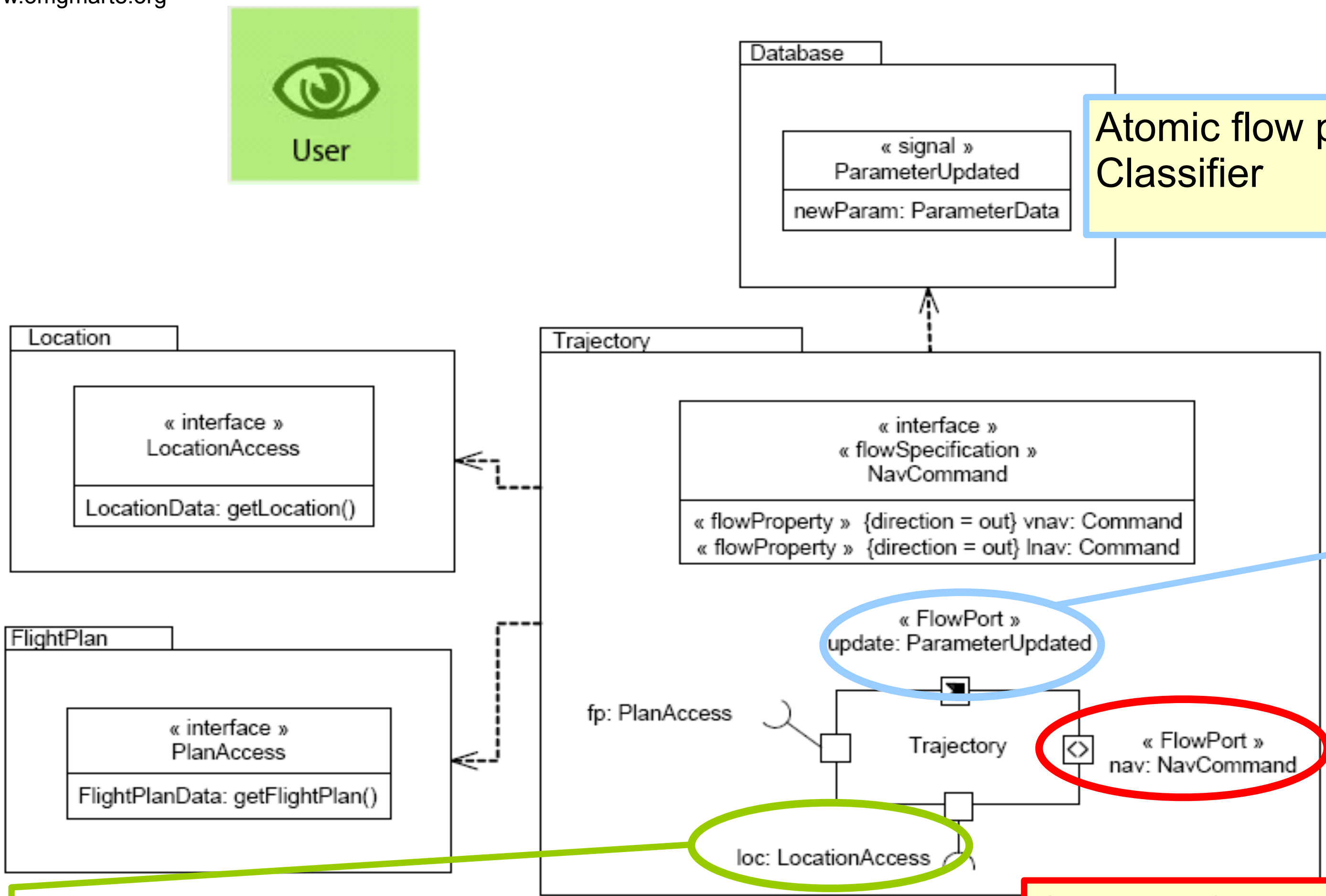
General Component Model

- **Introduced to cope with various component-based models**
  - SysML, Spirit, AADL, Lightweight-CCM, EAST-ADL2, Autosar
- **Does not imply any specific model of computation**
- **Relies mainly on UML structured classes, on top of which a support for SysML blocks has been added**
  - Atomic and non-atomic flow ports
  - Flow properties and flow specifications
- **But also providing a support for Lightweight-CCM, AADL and EAST-ADL2, Spirit and Autosar**

The MARTE GCM subprofile



Example of component definition

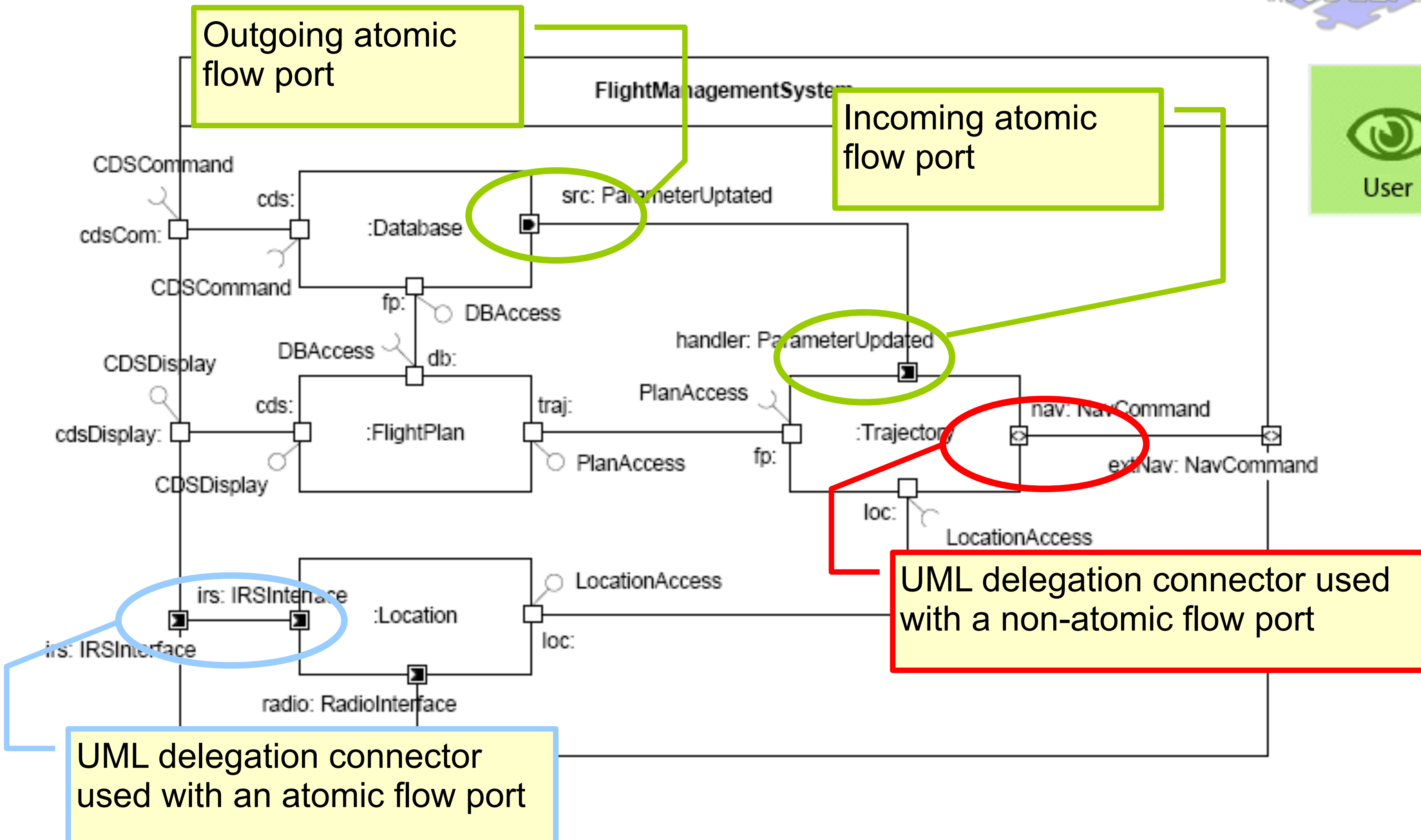


Atomic flow port typed by a Classifier

Complex flow port typed by a flow specification

Standard UML port typed by a class that uses the LocationAccess interface

Example of component usage



Agenda

- **Part 1**
  - Marte Overview
- **Part 2**
  - A component model for RT/E
- **Part 3**
  - **Non-functional properties modeling**

NFP tutorial Goals

- To give a “MARTE NFPs” background, yes, but also…

1. To know its main design rationales
2. To know its modeling capabilities/alternatives and to learn how and when to use them

Non-Functional Properties

An information of a modeled system, or system part, which is not related to its functional purpose
(e.g., latencies, memory size, power consumption)

- **Information used to:**
  - ➔ Static V&V: performance prediction, resource usage evaluation, …
  - ➔ Dynamic configuration: resource management, mode changes, …
- **Should include design process information:**
  - ➔ How information was obtained (estimations, measurement,…)?
  - ➔ How information was calculated (derived parameters, refinement)?
  - ➔ How accurate information is?

Previous Work

- **Main known approaches:**
  - ➔ NFR Framework: an approach to evaluate different quality goals and their interference.
  - ➔ CQML (Aagedal): language for expressing qualities in contract-aware components (no semantic link to execution model)
  - ➔ QCCS: it enhances CQML by providing refinement capabilities and a link to a basic execution model (operation, event identification)

- **OMG approaches:**
  - ➔ SPT Profile: Tag Value Language (TVL) for expressing mathematical, logical and basic time expressions.
  - ➔ QoS Profile (based on CQML): QoS value qualifiers, required an offered constraints (assume/guarantee component interfaces), catalog approach.

The MARTE's NFP Modeling Framework (NFPs Profile + VSL)

A set of extensions to specify semantically rich non-functional annotations

1. **NFPs sub-profile → based on the QoS Profile:**
   - ➔ Measurements: magnitude + unit (e.g., energy, data size, duration)
   - ➔ Qualifiers: Value source, statistical measure, value precision,…

2. **Value Specification Language (VSL) → based on TVL and OCL:**
   - ➔ Mathematical expressions (arithmetic, logical, …)
   - ➔ Time expressions (delays, periods, trigger conditions,…)
   - ➔ Variables: placeholders for unknown analysis parameters.

www.omgmarte.org

NFP Framework: Basic Description

The NFPs Profile

- **Three main stereotypes:**
  - → Nfp, Nfptype, NfpConstraint, Unit

- **A predefined library of Units and NfpTypes:**
  - → Power, Frequency, DataSize, DataTxRate, Duration, BoundDuration,…
  - → A set of generic qualifier for these NFP Types
  - → A placeholder for “expressions” in addition to the actual “value”.

- **Three mechanism to annotate NFP values:**
  1. Tagged Values
  2. InstanceSpecification Slots
  3. Constraints

Extending UML Expressiveness for NFPs

www.omgmarte.org

CAN Bus with Pure UML

CAN_Bus

transMode= transModeKind
speedFactor= Real
capacity= Integer
packetT= Real

can1: CAN_Bus

transMode= Half-Duplex
speedFactor= 0.8
capacity= 4
packetT= 64

CAN Bus with MARTE's NFP

CAN_Bus

transmMode: TransmModeKind
speedFactor: NFP_Real
capacity: NFP_DataTxRate
packetT: NFP_Duration

can1: CAN_Bus

transMode= Half-Duplex
speedFactor= (0.8, est)
capacity= (4, $capCan1, kHz, max, req)
packetT= (64, pckSize/capCan1, ms, calc)

User

Basic NFP Annotation Mechanism (Slots)

www.omgmarte.org

1) Declare NFP types

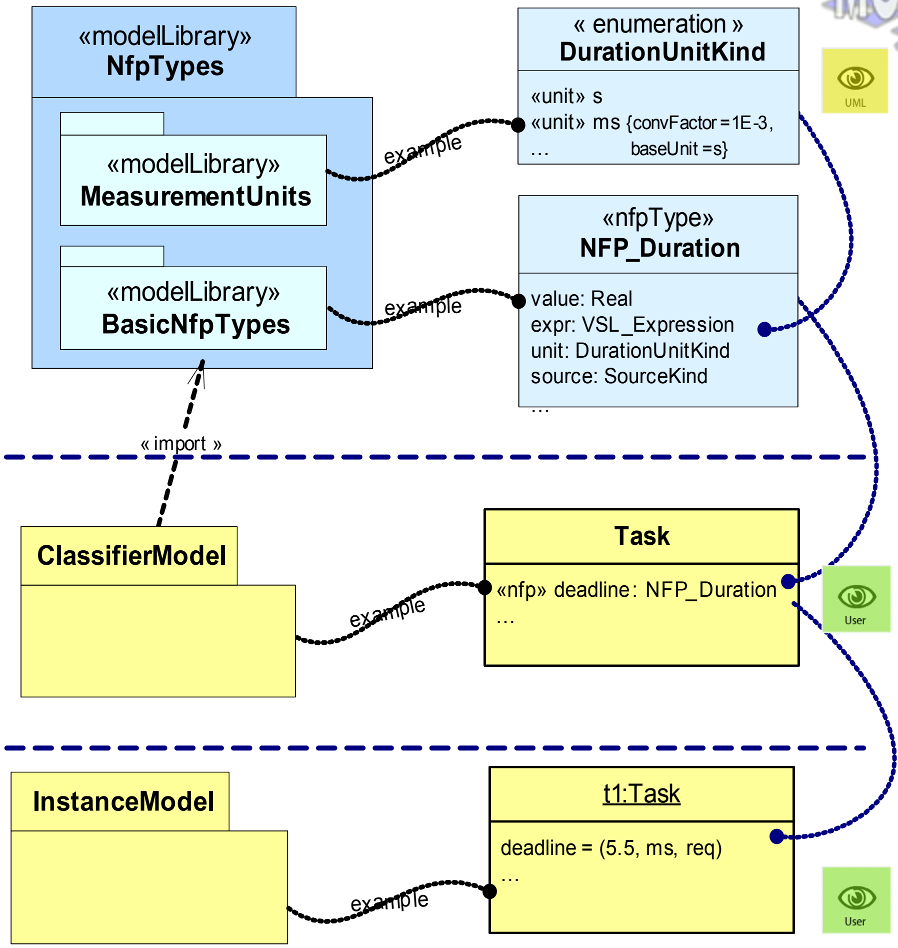
- Define measurement units and conversion parameters
- Define NFP types with qualifiers

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



The Value Specification Language

- **Formally defined (unlike TVL)**
  - ➔ Provide a metamodel based on the UML one (DataTypes & ValueSpecifications)
    - ➔ A subset of additional metaclasses are implemented as Stereotypes (DataTypes)
  - ➔ Expression metaclasses are extended as an orthogonal language.
    - ➔ MARTE proposes a grammar implementation
    - ➔ It may be implemented by a separated metamodel (e.g. based on Ecore)
- **A extended system of data types:**
  - ➔ Composite types: Tuples, Choice, Collection types
  - ➔ Subtypes: bounded subtype
- **A extended language for expressions:**
  - ➔ Primitives, Composites, Expression&Variables, Time Expressions.
- **Three mechanism to annotate NFP values:**
  - ➔ Tagged Values, InstanceSpecification Slots, and Constraints

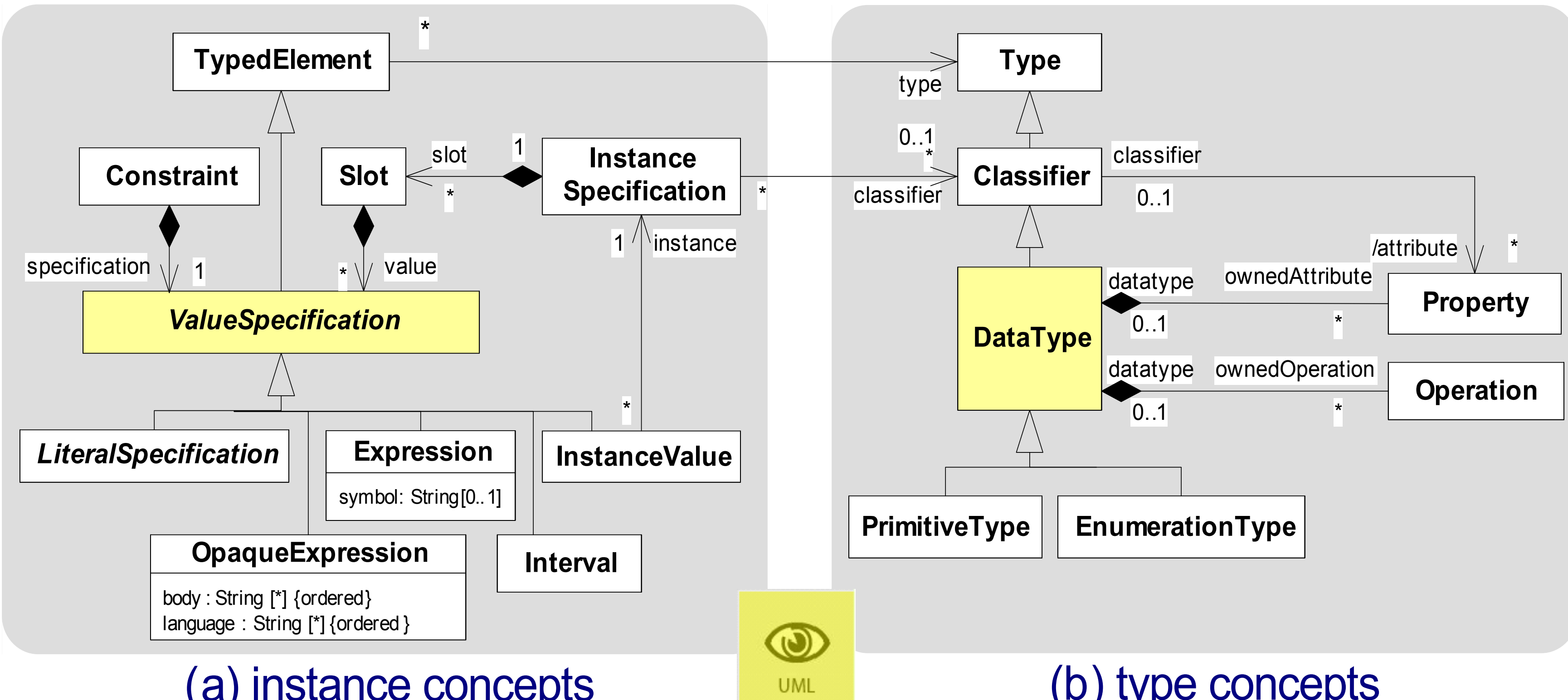
Basic Textual Expressions in VSL

- 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`

Design Rationales & Further Usage

UML-Related Concepts



(a) instance concepts

(b) type concepts

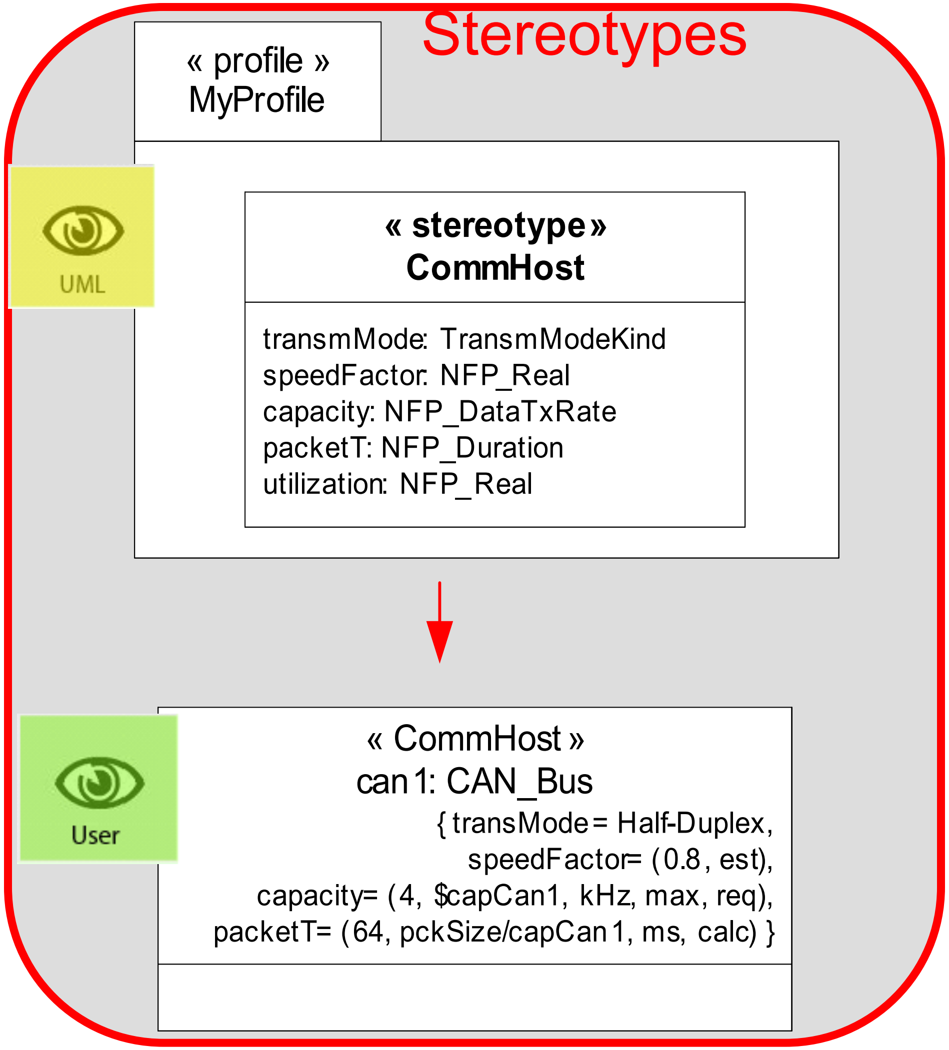
Some Design Choices

- **Why not the QoS Profile?**
  - Too complex annotation mechanism
    - Two stages to annotate model & models borrowed by InstanceSpecifications
  - But mainly… political reasons ;-)
  - ***Result → NFP stereotypes replace QoS ones***

- **Why not Stereotypes to extend UML expressions?**
  - As we intend to annotate NFP values in Tagged Values, we don't want to stereotype information within TaggedValues
  - We don't want to confuse users by mixing M1 and M2 levels
  - ***Result → Qualifiers in NfpTypes & a new textual language: VSL***

Annotating NFPs in Tagged Values: Ex.

CAN Bus annotated with Stereotypes



Annotating NFPs in Tagged Values

1) Declare NFP types

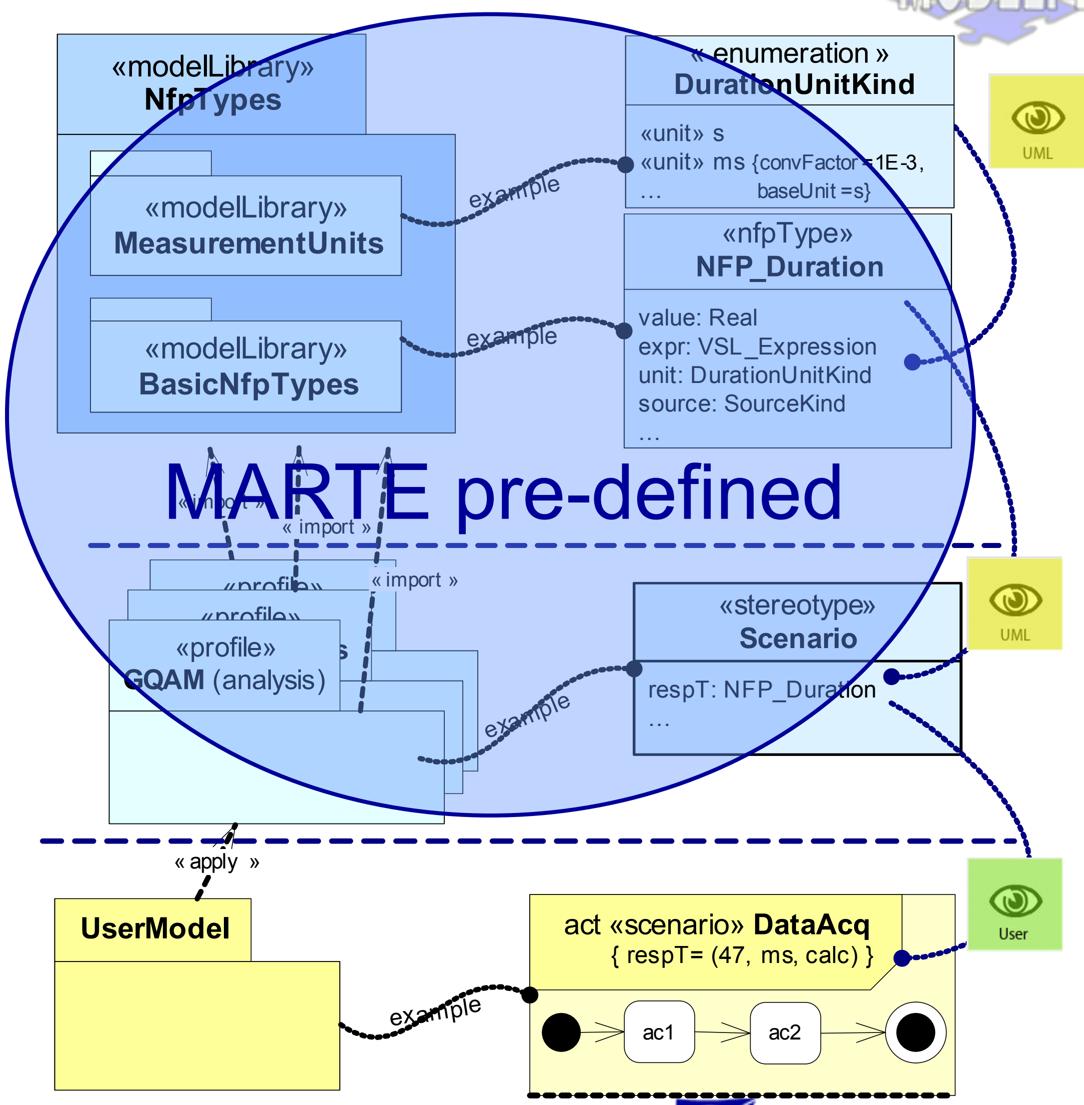
- Define measurement units and conversion parameters
- Define NFP types with qualifiers

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 Constraints

1) Declare NFP types

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

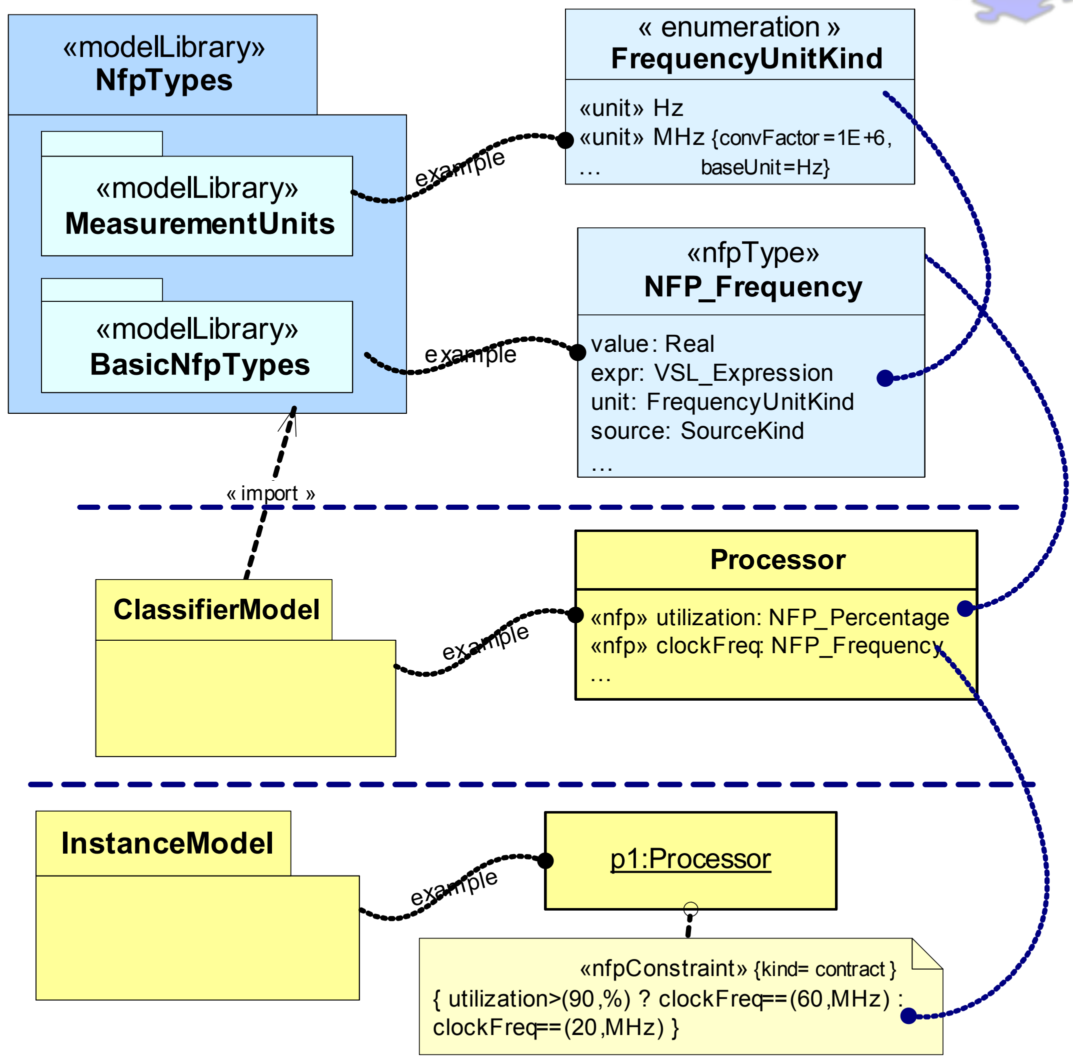
2) Declare NFPs

-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



Some Design Choices (2)

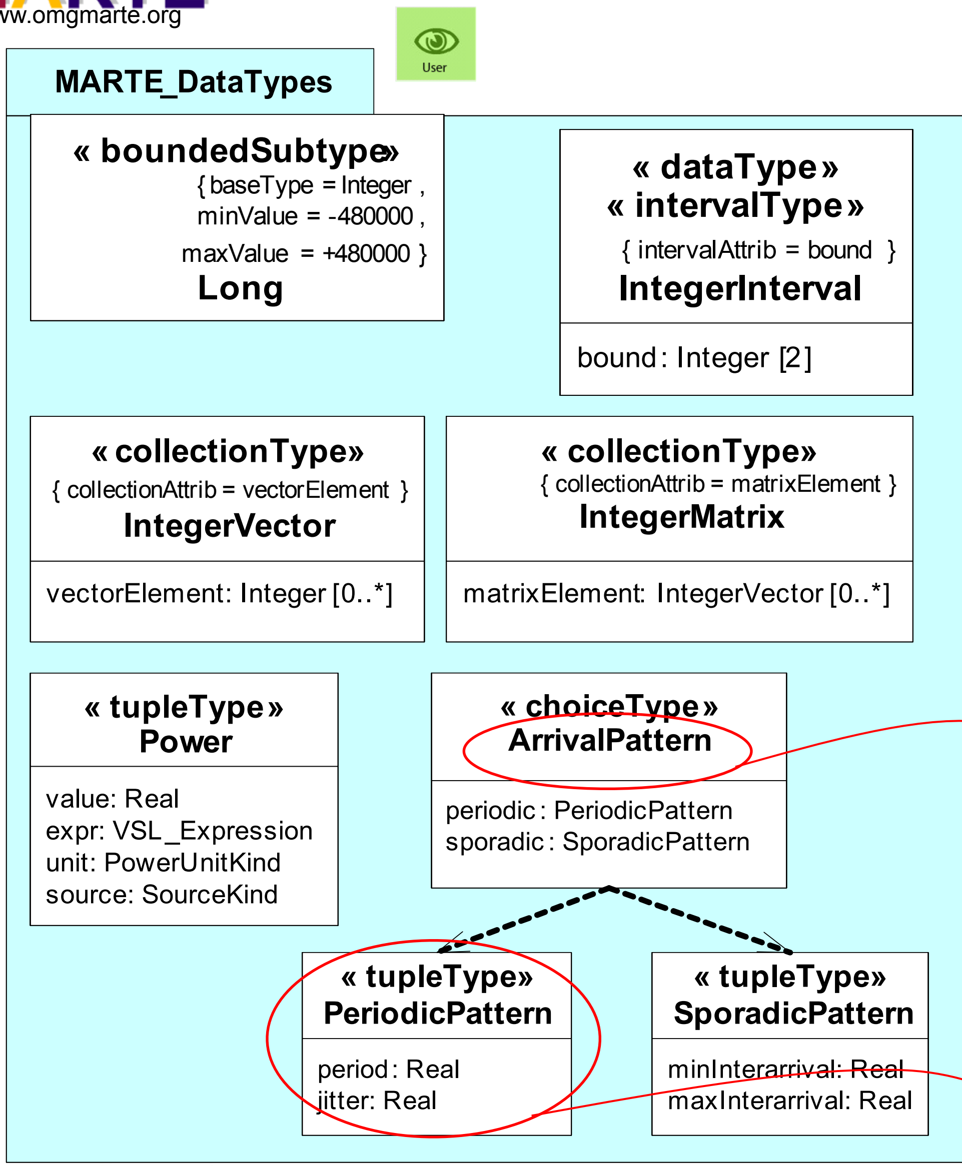
- **Why not OCL instead of VSL?**
  - Too hard to learn for end users
  - No time expressions
  - ***Result → VSL reuse OCL syntax, but the metamodel is closer to UML one***

- **VSL take into account the ISO Standard: General Purpose DataTypes**
  - It allows to map the VSL syntax to standard programming languages (e.g., C, Pascal)

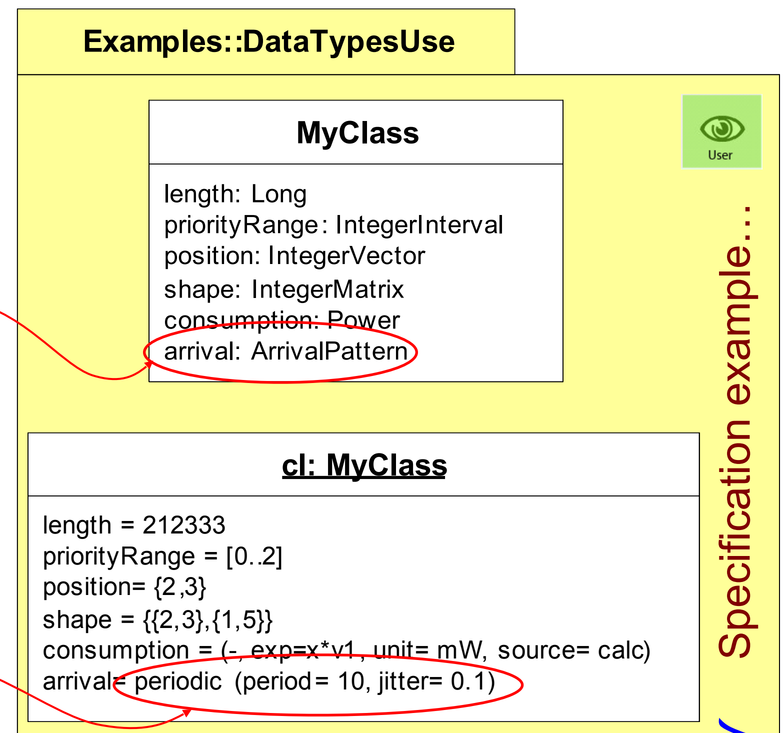
VSL Extended Data Types

- BoundedSubtype
- IntervalType
- CollectionType
- TupleType
- ChoiceType

Declaration example…



Specification example…

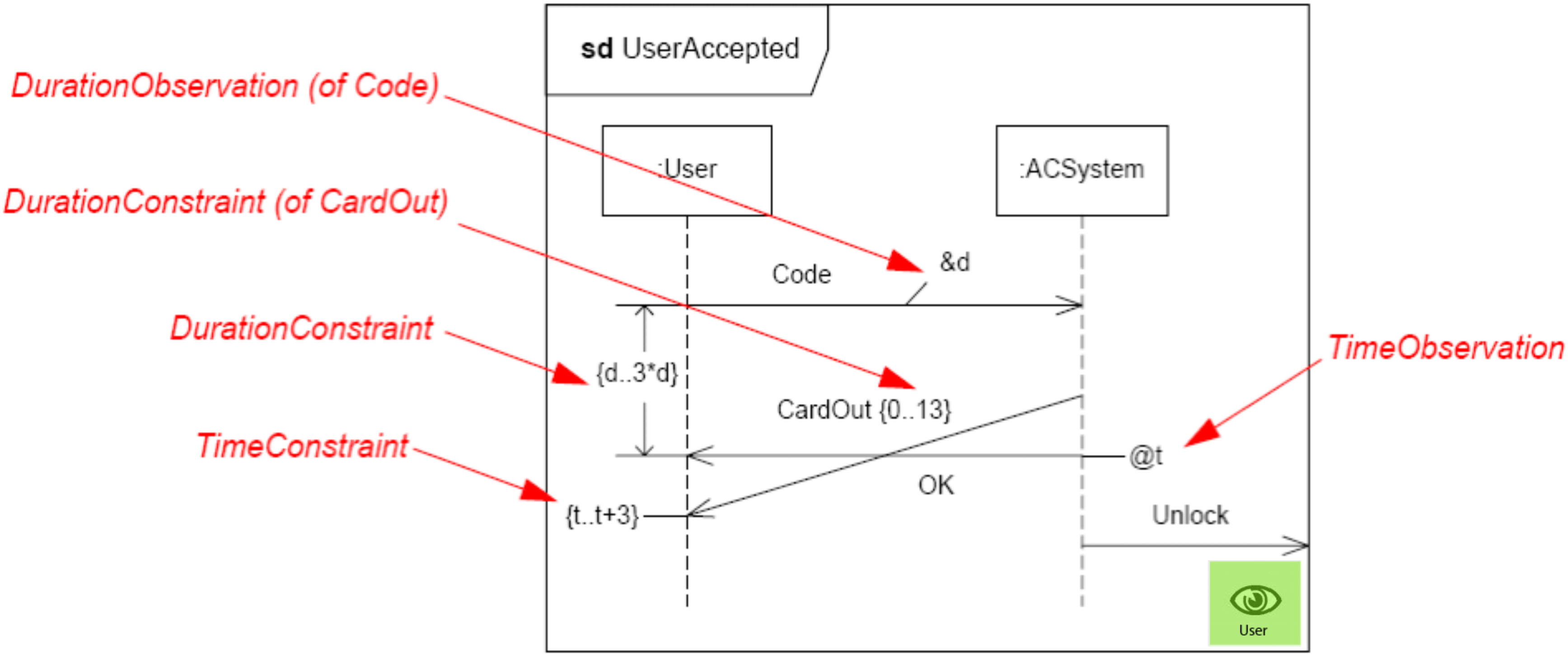


UML (native) Observation Concept

An time observation is a reference to a time instant during an execution.

An duration observation is a reference to a time interval during an execution.

Specification example in Sequence diagrams…



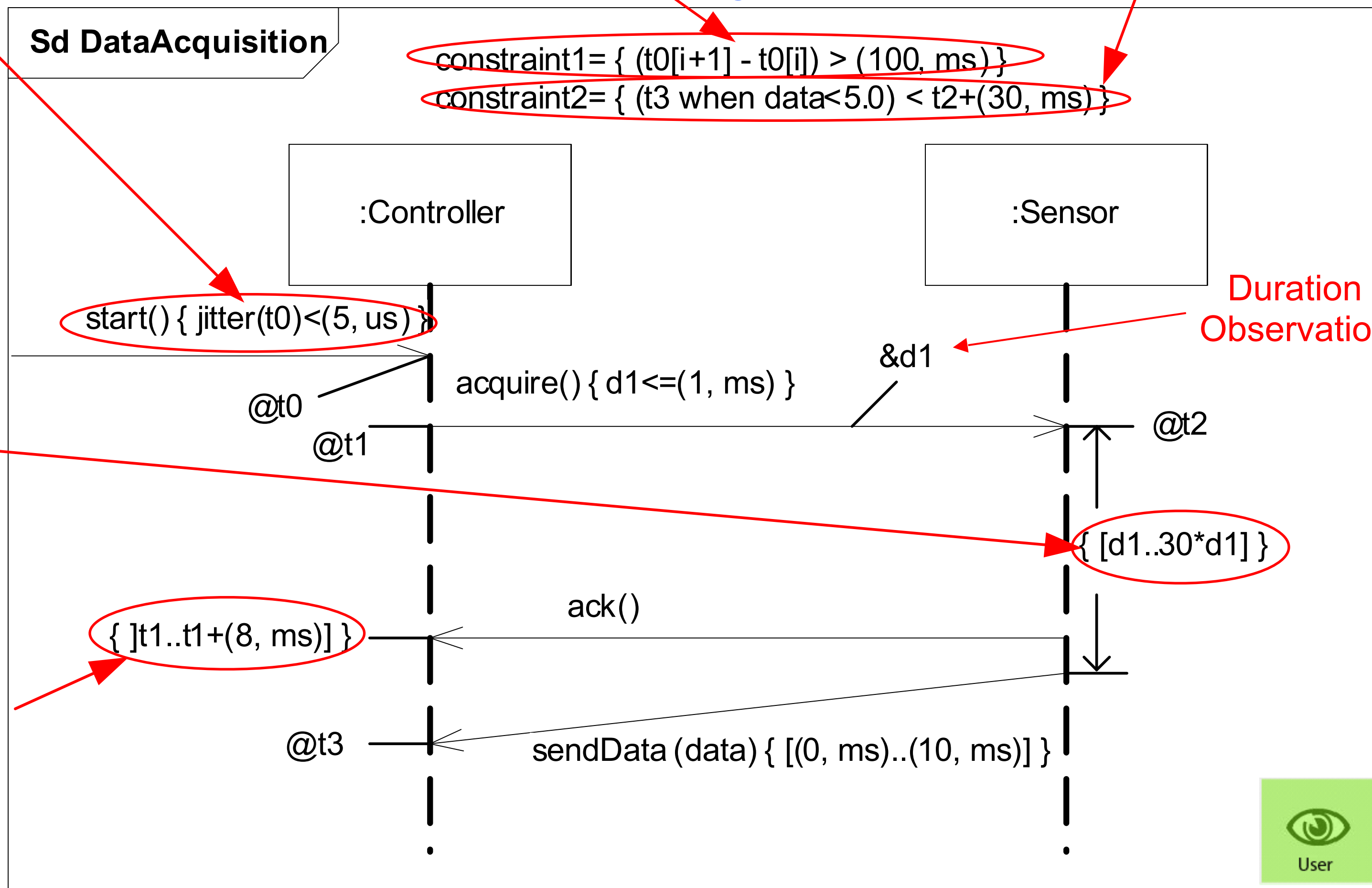
Time Expressions with VSL

Jitter constraint

Duration expression between two sucessive occurrences

Constraint in an observation with condition expression

Specification example in Sequence diagrams…



Extended duration intervals with bound « [ ] » specification

Duration Observation

Instant Interval Constraint

Summary of NFP Modeling Framework

1. Usability vs. Flexibility:
   - ➔ Three annotation mechanisms: Stereotypes, Properties and Constraints
   - ➔ Stereotypes: predefined NFPs (e.g., end-to-end latency, processor utilization)
   - ➔ Properties & Constraints: user-specific NFPs (but still unambiguously interpreted)
2. Synthesis of best modeling practices…
   - ➔ Reuse OCL constructs: grammar for values and expressions
   - ➔ Formally defined by abstract (metamodel) and concrete (grammar) syntaxes→ Can be implemented as non-UML based language
   - ➔ VSL supports time expressions (occurrence index, jitters,...)