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

Component-based paradigms in the RTE domain

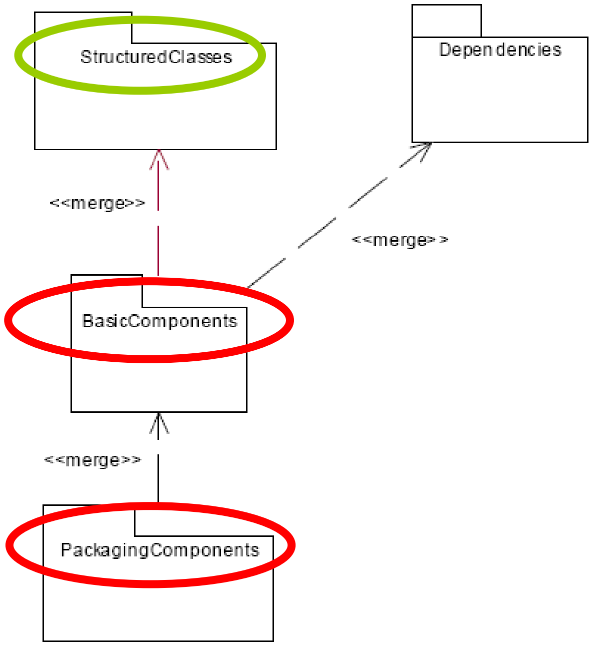
- **Component architectures are increasingly used in RTE execution platforms**
  - Need for manageable and reusable pieces of software
  - Key examples: Lightweight-CCM, SCA, Autosar

- **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

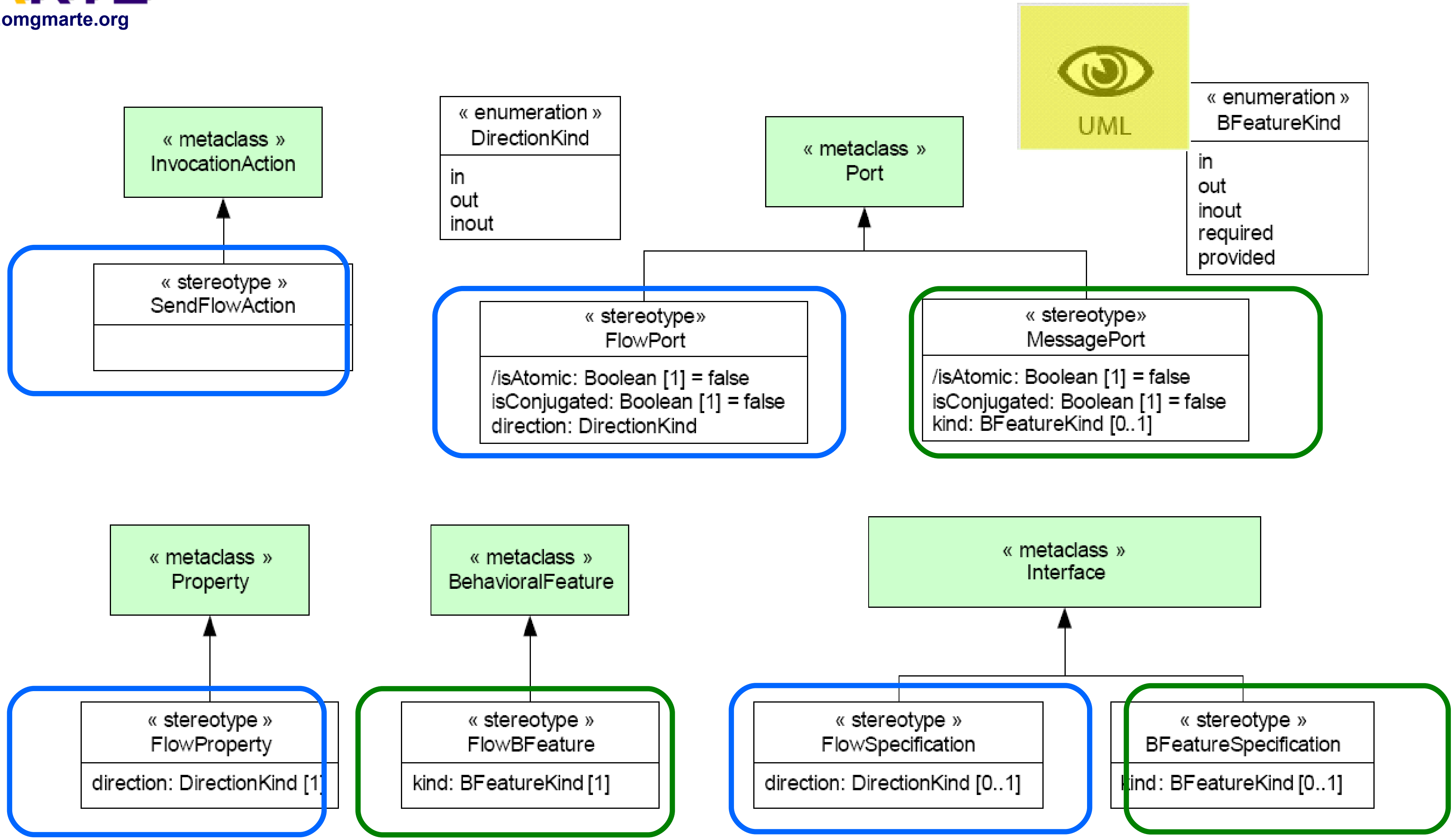


- In any case, no support for flow-oriented communications

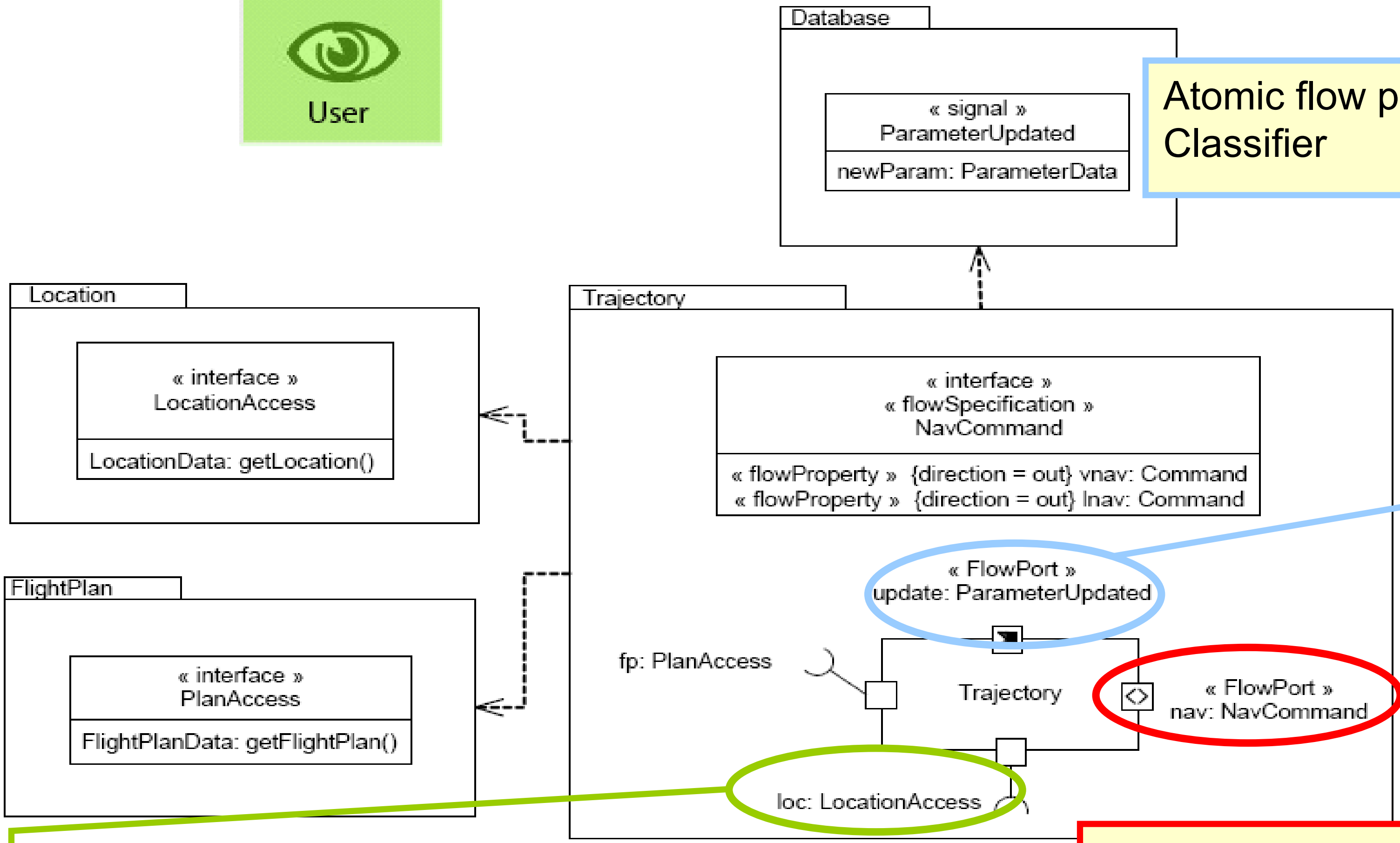
General Component Model

- **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



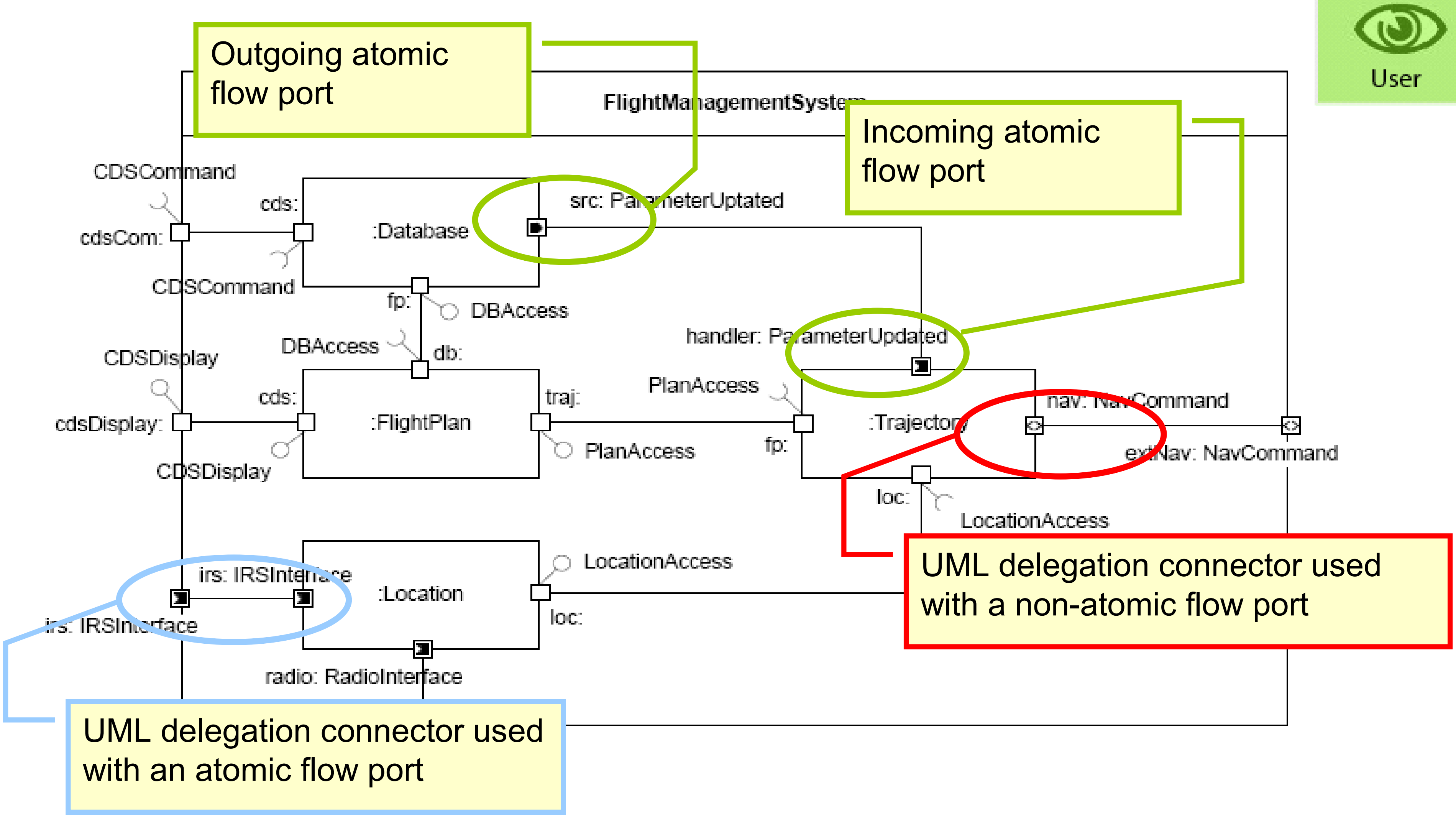
Atomic flow port typed by a Classifier

Complex flow port typed by a flow specification

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

Example of component usage

User



Outgoing atomic flow port

Incoming atomic flow port

UML delegation connector used with a non-atomic flow port

UML delegation connector used with an atomic flow port

RTE Model of Computation and Communication

- **High-level modeling concepts for RT/E design**
  - Qualitative aspects
    - E.g. concurrency and behavior
  - Quantitative aspects as real-time feature
    - E.g. deadline or period
- **Allows expressing real-time constraints on component interfaces and connectors**
  - Applicable whether component are active or passive
- **For active components, introduces specific models of computation**
  - Currently, active objects (e.g. Rhapsody, Rose RT, ACCORD)
  - Alternative MoCC can be defined using the MARTE foundations

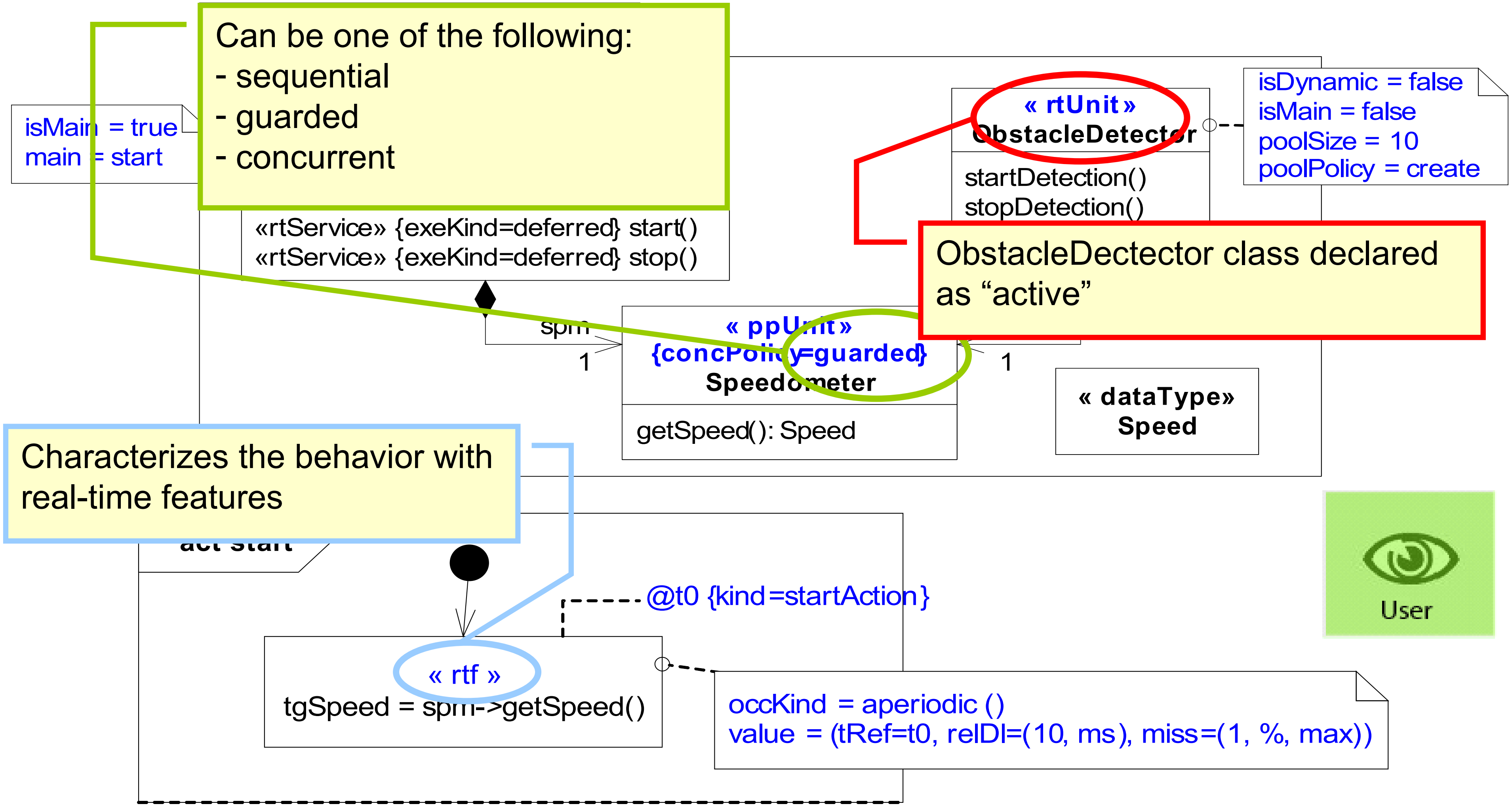
RTE Model of Computation and Communication

- **Provides high-level concepts for modeling qualitative real-time features on classes / structured classes / components**
  - Real-Time Unit (RTUnit)
    - Generalization of the Active Objects of the UML 2
    - Owns at least one schedulable resource
    - Resources are managed either statically (pool) or dynamically
    - May have operational mode description (similar to AADL modes)
  - Protected Passive Unit (PPUnit)
    - Generalization of the Passive Objects of the UML2
    - Requires schedulable resources to be executed
    - Supports different concurrency policies (e.g. sequential, guarded)
    - Policies are specified either locally or globally
    - Execution is either immediateRemote or deferred

RTE Model of Computation and Communication (cont'd)

- **Provides high-level concepts for modeling quantitative real-time features on classes / structured classes / components**
  - Real-Time Behavior (RtBehavior)
    - Message Queue size and policy bound to a provided behavior
  - Real-Time Feature (RTF)
    - Extends UML Action, Message, Signal, BehavioralFeature
    - Relative/absolute/bound deadlines, ready time and miss ratio
  - Real-Time Connector (RteConnector)
    - Extends UML Connector
    - Throughput, transmission mode and max blocking/packet Tx time

Usage examples of the RTEMoCC extensions



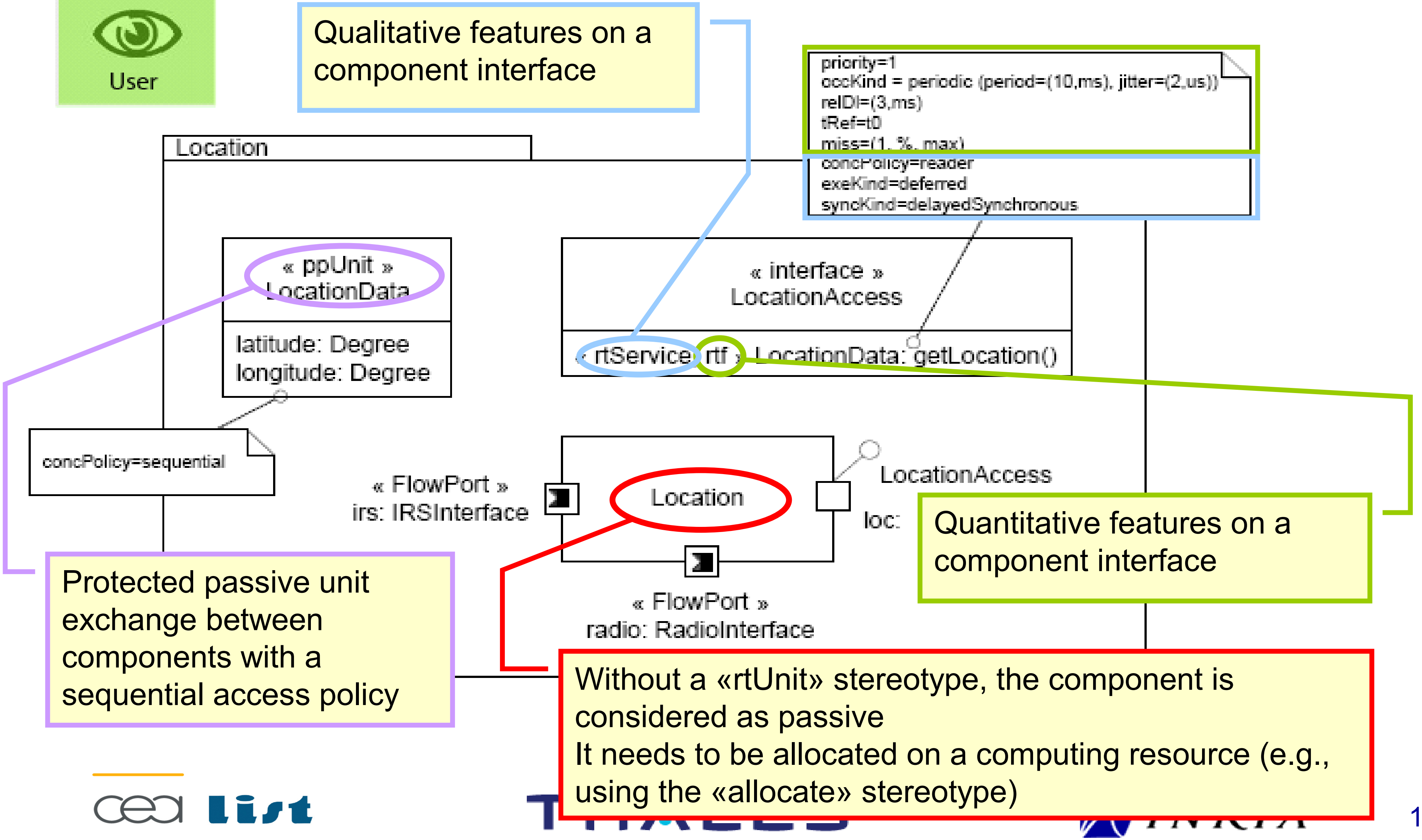
Can be one of the following:
- sequential
- guarded
- concurrent

ObstacleDectector class declared as “active”

Characterizes the behavior with real-time features

Modeling real-time features of components

User



Qualitative features on a component interface

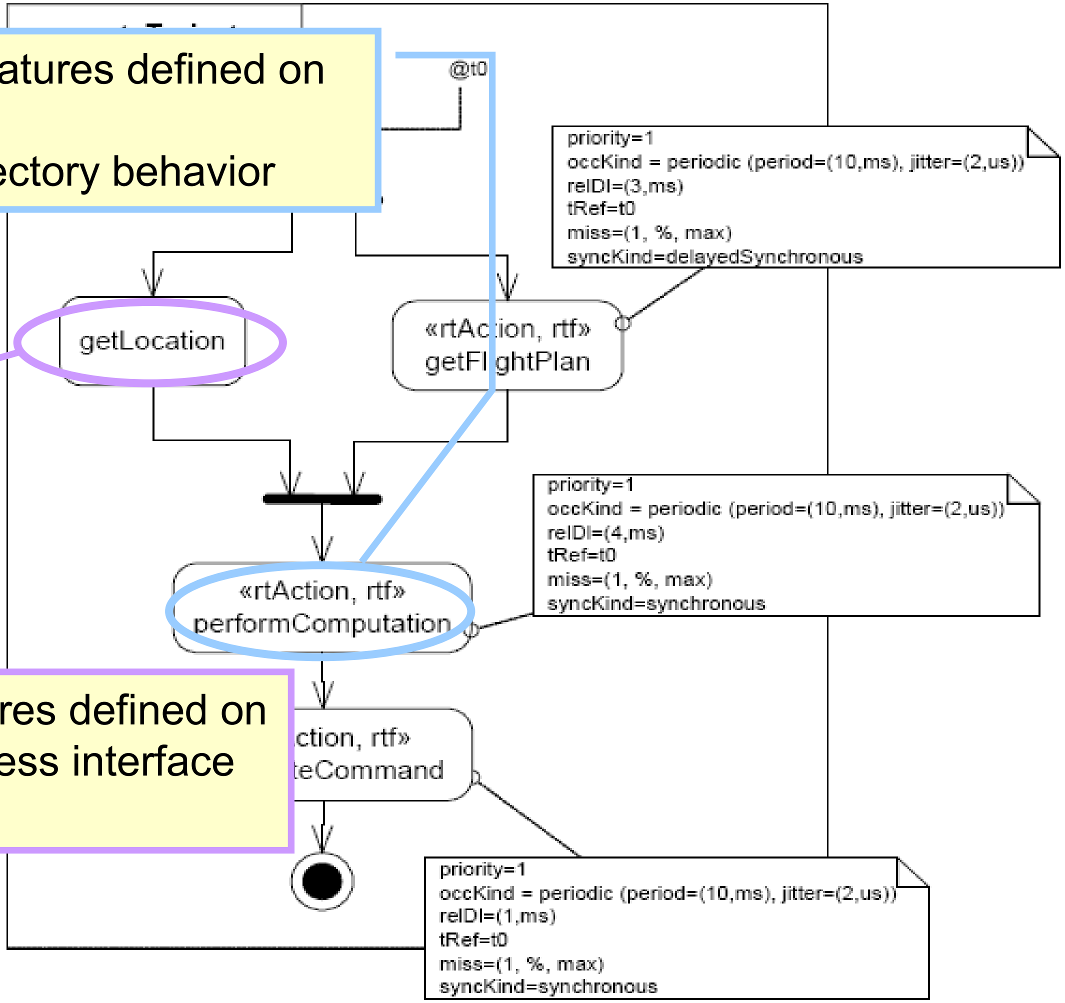
Quantitative features on a component interface

Protected passive unit exchange between components with a sequential access policy

Without a «rtUnit» stereotype, the component is considered as passive
It needs to be allocated on a computing resource (e.g., using the «allocate» stereotype)

Modeling real-time features of components (cont’d)

User



priority=1
occKind = periodic (period=(10,ms), jitter=(2,us))
relDl=(3,ms)
tRef=t0
miss=(1, %, max)
syncKind=delayedSynchronous

priority=1
occKind = periodic (period=(10,ms), jitter=(2,us))
relDl=(4,ms)
tRef=t0
miss=(1, %, max)
syncKind=synchronous

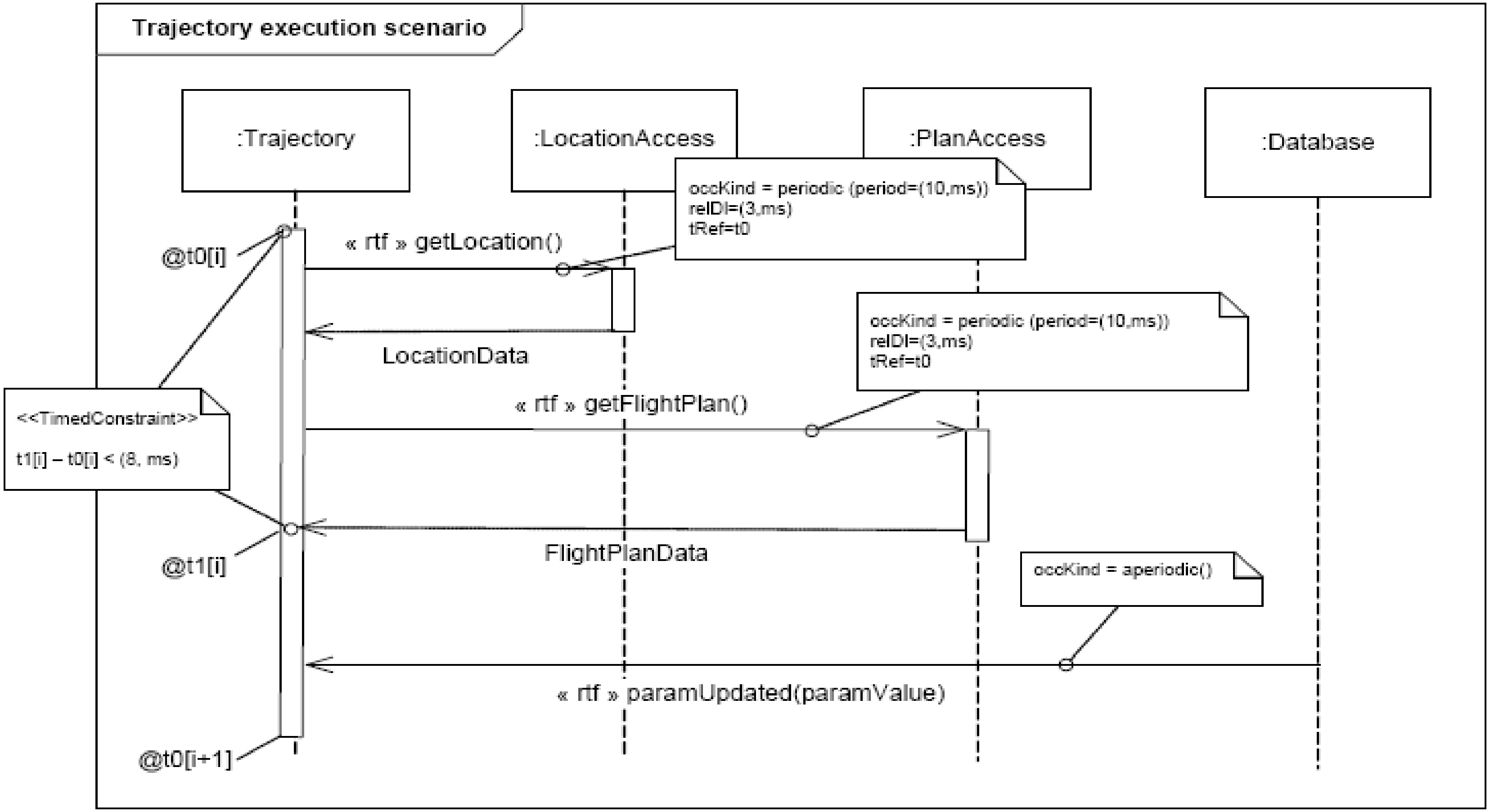
priority=1
occKind = periodic (period=(10,ms), jitter=(2,us))
relDl=(1,ms)
tRef=t0
miss=(1, %, max)
syncKind=synchronous

Qualitative features defined on actions of the computeTrajectory behavior

Qualitative features defined on the LocationAccess interface apply

Modeling real-time features of components (cont'd)

User



Extensibility

- **All models of computation in the RTE domain not explicitly addressed by MARTE**
- **MARTE foundations (NFP, Time, GRM) allow third-parties to specify other model of computations that rely on the same semantic basis**
  - Allows one to use MARTE features along with this user-defined MoCC