TUTORIAL ON THE UML

François Terrier

CEA Saclay Nano-INNOV - Institut CARNOT CEA LIST, DILS/LISE
Point Courrier n° 174 - 91 191 Gif sur Yvette CEDEX

francois.terrier@cea.fr

www.omg.org

www.eclipse.org/papyrus

- **Language = syntax + semantics**
  - Syntax = rules by which language elements (e.g., words) are assembled into expressions (e.g., phrases, clauses)
  - Semantics = rules by which syntactic expressions are assigned meanings

➔ **Générique et Expressif**

➔ **Syntaxe et sémantique**

➔ Notion de syntaxe abstraite :
on dissocie les concepts des choix de représentation
(apparence du langage)

Rules for naming, scoping, visibility, integrity
+ execution (limited)

Example of semantic rule: Class [1]

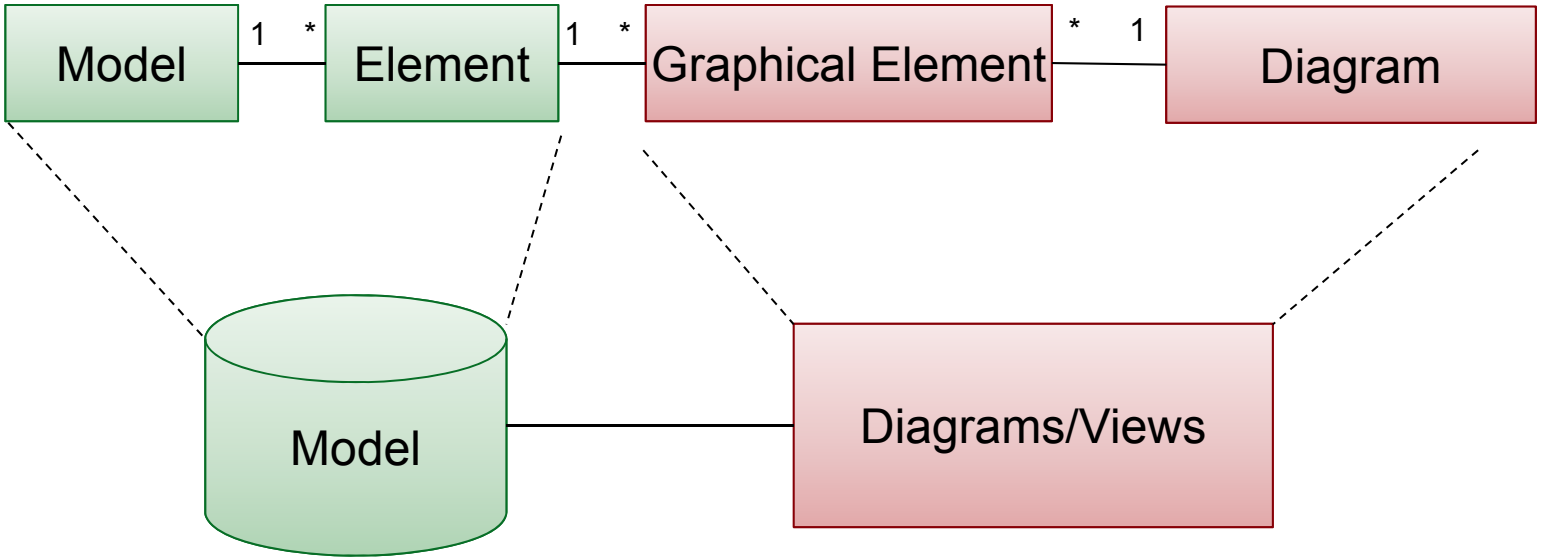
- English: If a Class is concrete, all the Operations of the Class should have a realizing Method in the full descriptor.

- OCL:
```
not self.isAbstract implies self.allOperations->
    forAll (op | self.allMethods->
        exists (m | m.specification-> includes(op)))
```

Example of syntactic rules: Class

- Basic Notation:
  - e.g. “A class is drawn as a solid-outline rectangle with three compartments separated by horizontal lines.”

- Presentation Option
  - e.g. “Either or both of the attribute and operation compartments may be suppressed.”



- When you delete an element from the model you delete all its graphical elements from the diagram.
- When you delete a graphical element, you DO NOT necessarily delete the corresponding element in the model.

A unique formalism for any application type

- Data base, embedded systems, multimedia, information system…

! ***But UML stay at the language level***

→ ***it does not propose any development process/method***

- *nor concerning development task organisation*
- *or concerning responsability distribiution*
- *or related to usage rules*

View point synthesis

- **Static and structural**
- **Dynamic and behavioral**
- **Fonctionnal**

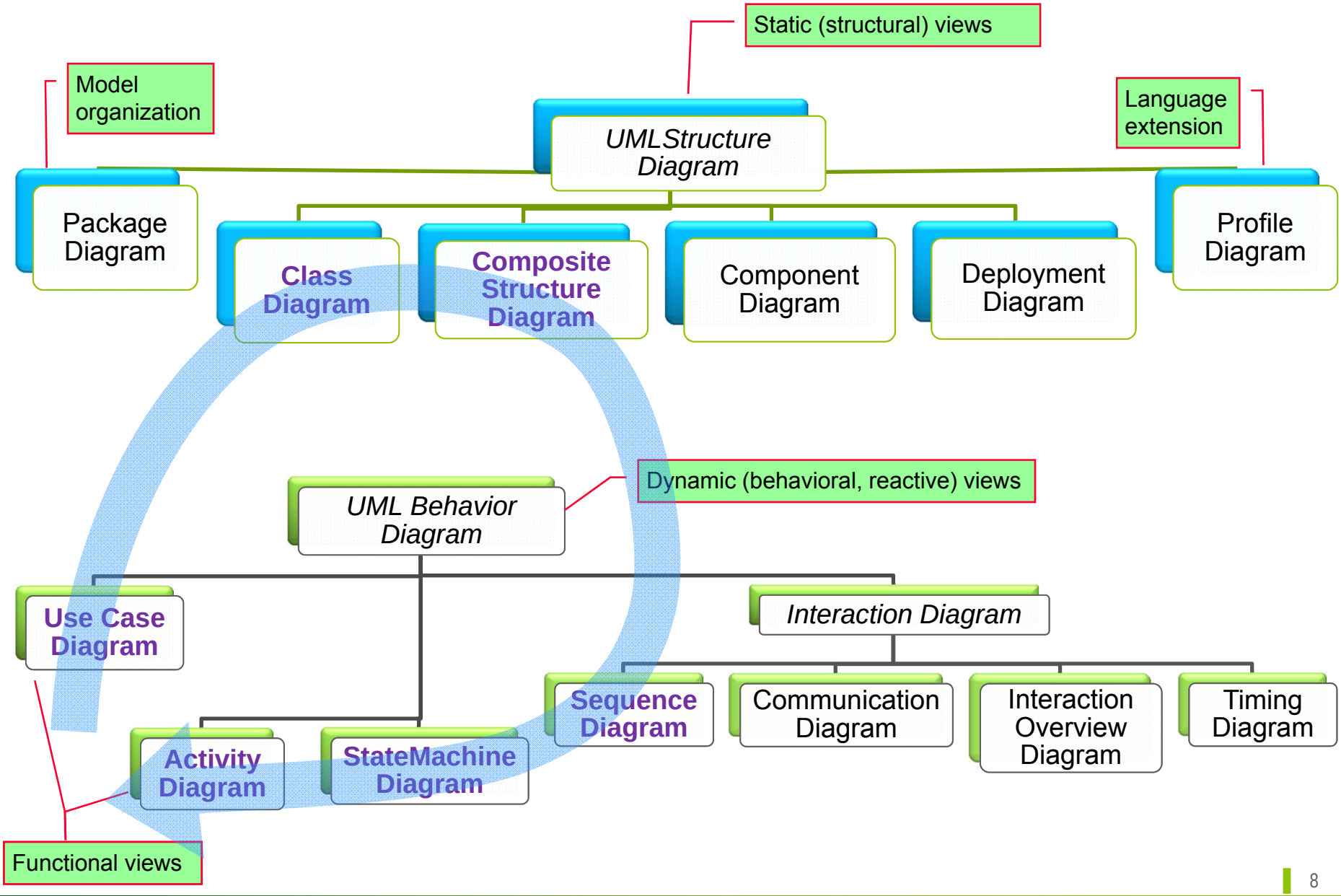
An incremental approach

- **From analysis to implementation through design with iterative refiment steps using the same formalism**
  - No language discontinuity
  - Possibility of continuous tool chain

- **UML is a notation, not a method**
- **UML is adequat for all the object methods**
- **UML is a modeling language almost but not necessarly object oriented…**
- **UML is free and public**

UML is the standard notation to build object models, architectures and to describe behaviors

UML DIAGRAMS



Synthesis of several view points

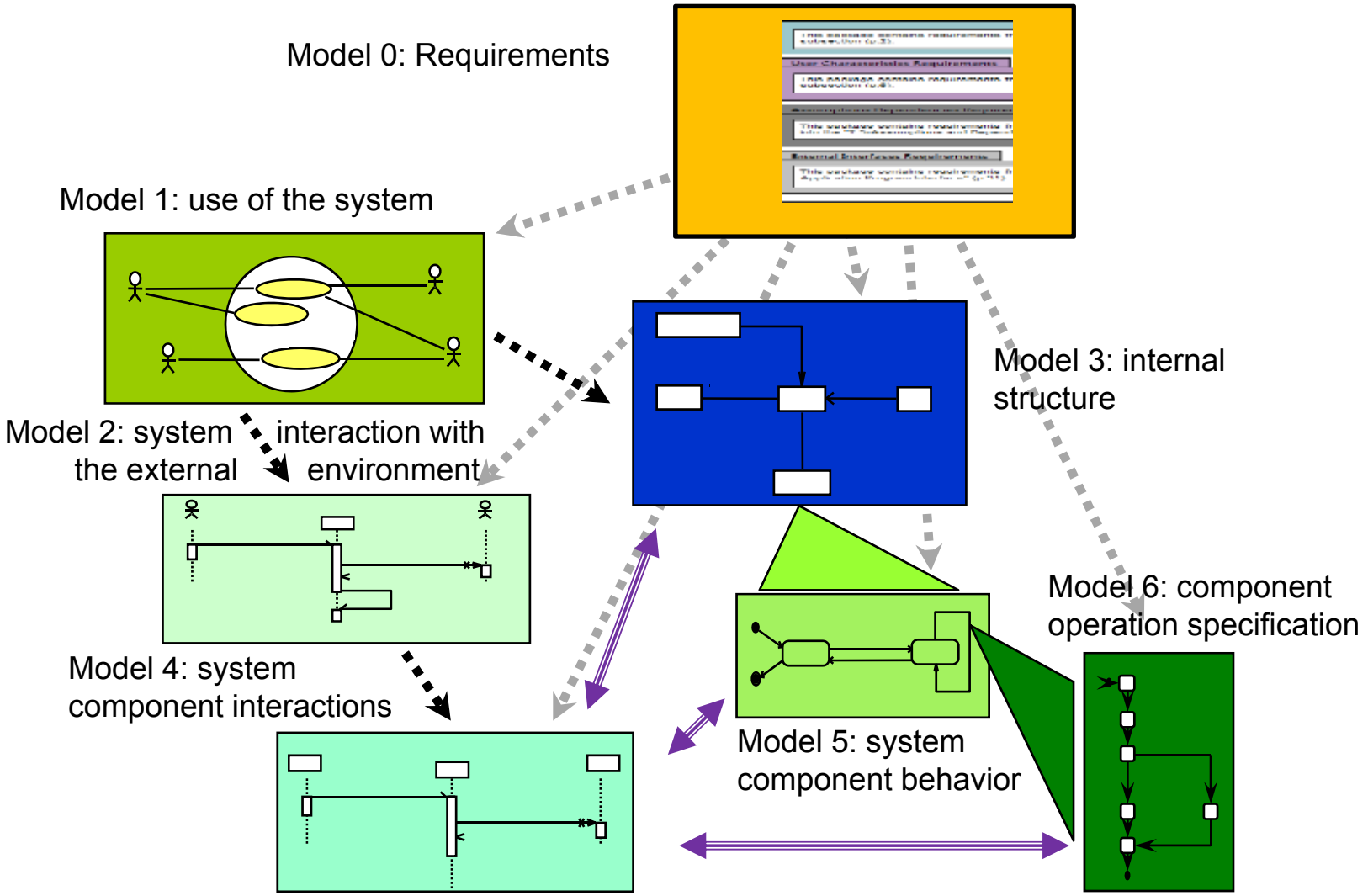
- Fonctionnal: usage and algorithms
- Static: structure
- Dynamic: reactive behaviour and interactions

➔ *Ensure the consistency among the view points*

Incremental approach of the development process from problem analysis to system implementation through iterative refinements of the system model (or set of models) using the same formalism!

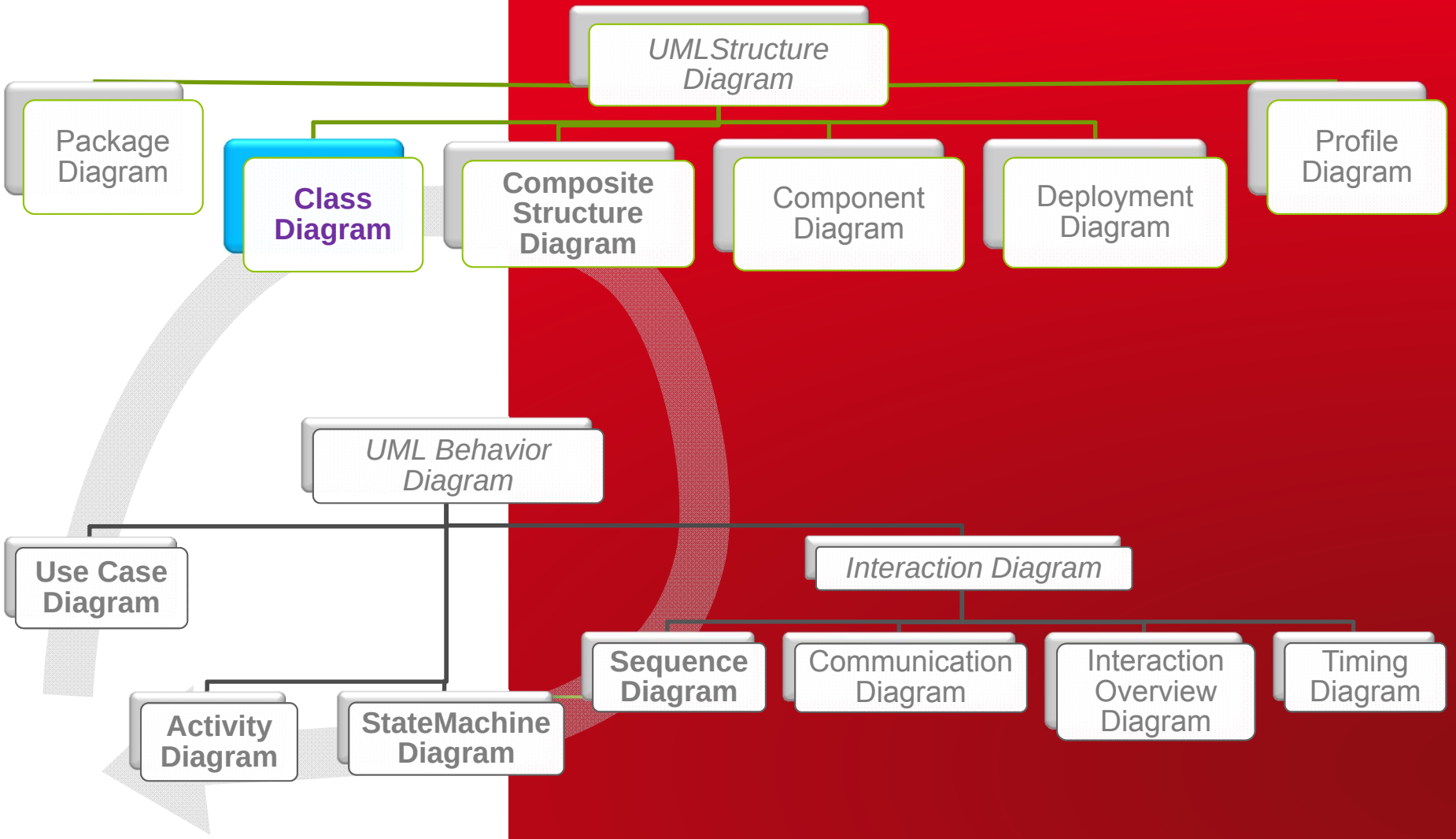
- No language discontinuity
- Possibility of continuity among the tools

Complementary models for complementary views of a complete model in an incremental approach



- **Consistency rules to ensure non ambiguous modeling**
- **Formal analysis of the models becomes possible**

UML Class Diagrams



WHAT IS FOR THE STRUCTURAL DIAGRAM

A structural model is a view of a system that emphasizes the structure of its elements: here its objects, including their classifiers, relationships, attributes and operations

FEATURES OF THE STRUCTURAL DIAGRAM

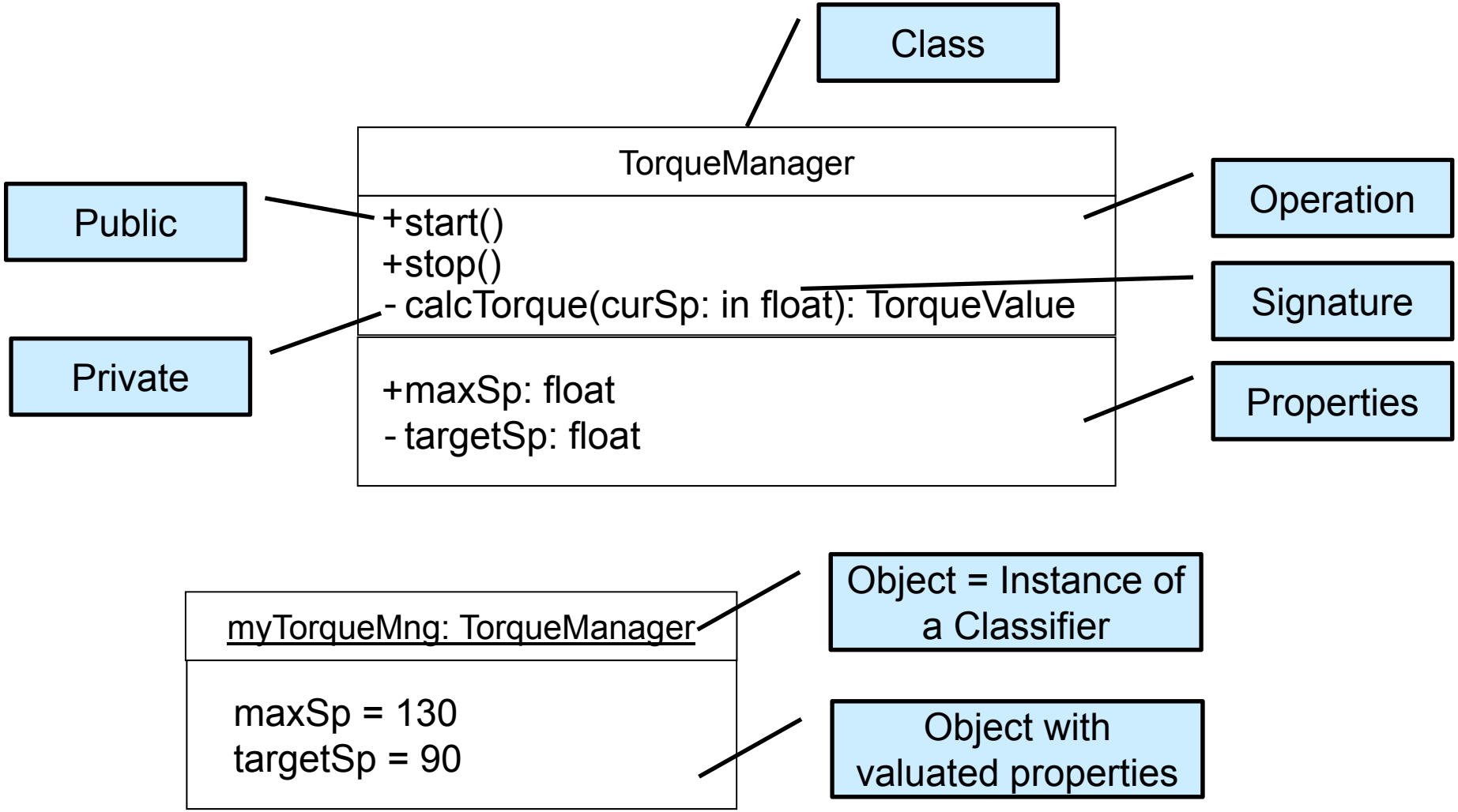
Core elements are:

- Class is a description of a set of objects that share the same attributes, operations, methods, relationships and semantics;
- Interface is a named set of operations that characterize the behavior of an element;
- Package is a way to organize the models into parts.

Relationship elements are:

- Association representing a structural relation
- Generalization representing a conceptual abstraction relation
- Dependency representing a technical link relation

CLASSIFIER


Class
TorqueManager
+start()
+stop()
- calcTorque(curSp: in float): TorqueValue
+maxSp: float
- targetSp: float
Public
Private
Operation
Signature
Properties
myTorqueMng: TorqueManager
maxSp = 130
targetSp = 90
Object = Instance of a Classifier
Object with valuated properties

RELATIONSHIP - ASSOCIATION

Association links communicating classes (~ message support medium)

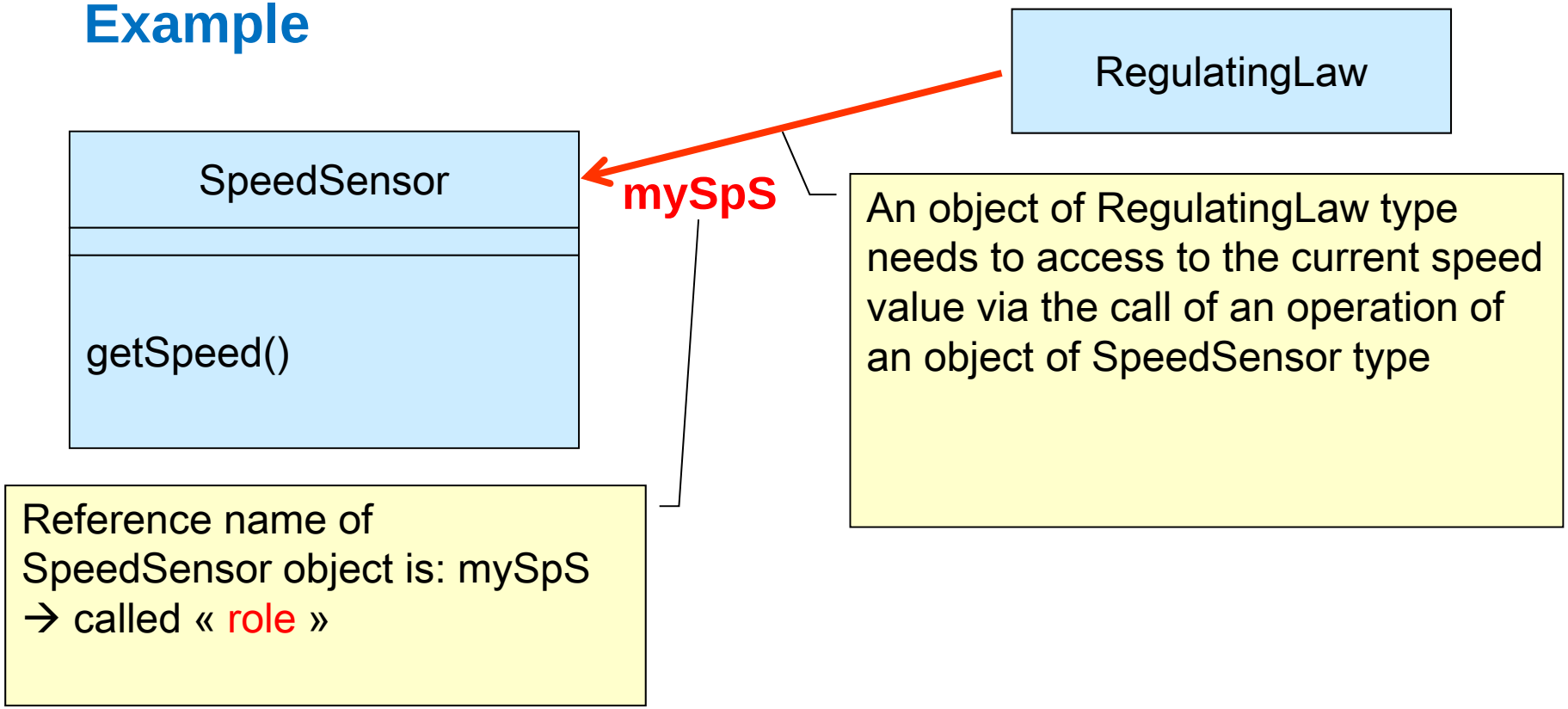
- **Multiplicity**
- **Role names**
- **Association name**
- **Navigabiltiy**

Special forms of association

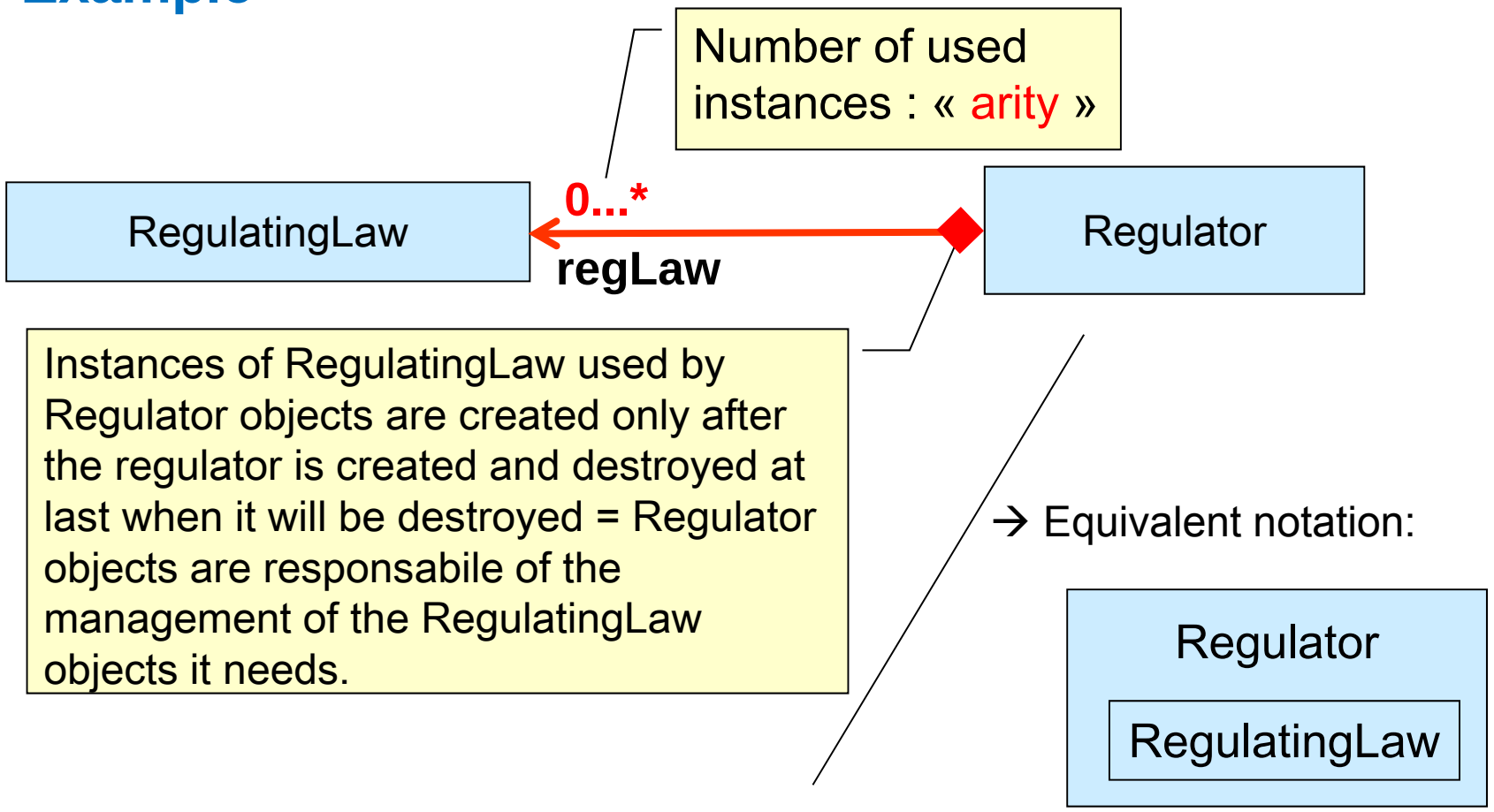
- **Aggregation specifies a whole-part relationship between the container and the contained parts**
  - contained parts may exist independently of their container
- **Composition denotes a strong ownership**
  - life of contained objects is dependent of the container life

It denotes the delegation of some subprocessing to other objects…

Example

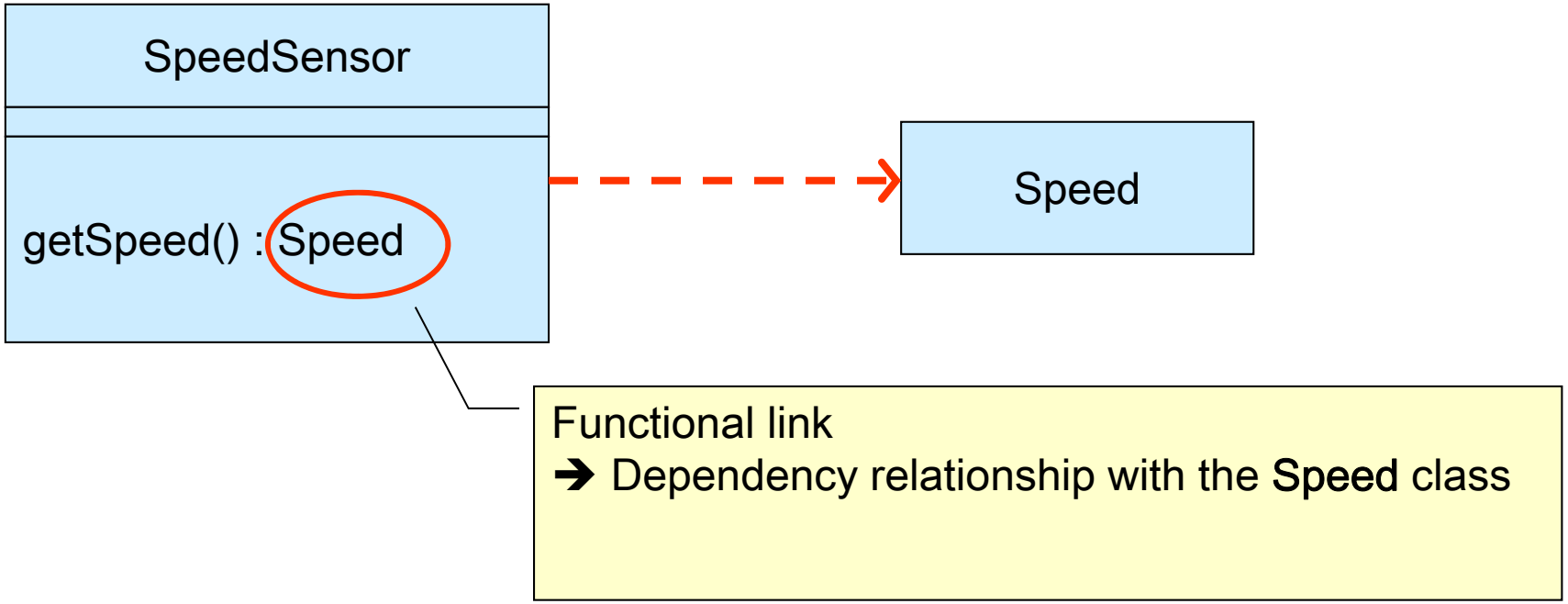


Example


Number of used instances : « arity »
RegulatingLaw
0...*
regLaw
Regulator
Instances of RegulatingLaw used by Regulator objects are created only after the regulator is created and destroyed at last when it will be destroyed = Regulator objects are responsabile of the management of the RegulatingLaw objects it needs.
→ Equivalent notation:
Regulator
RegulatingLaw

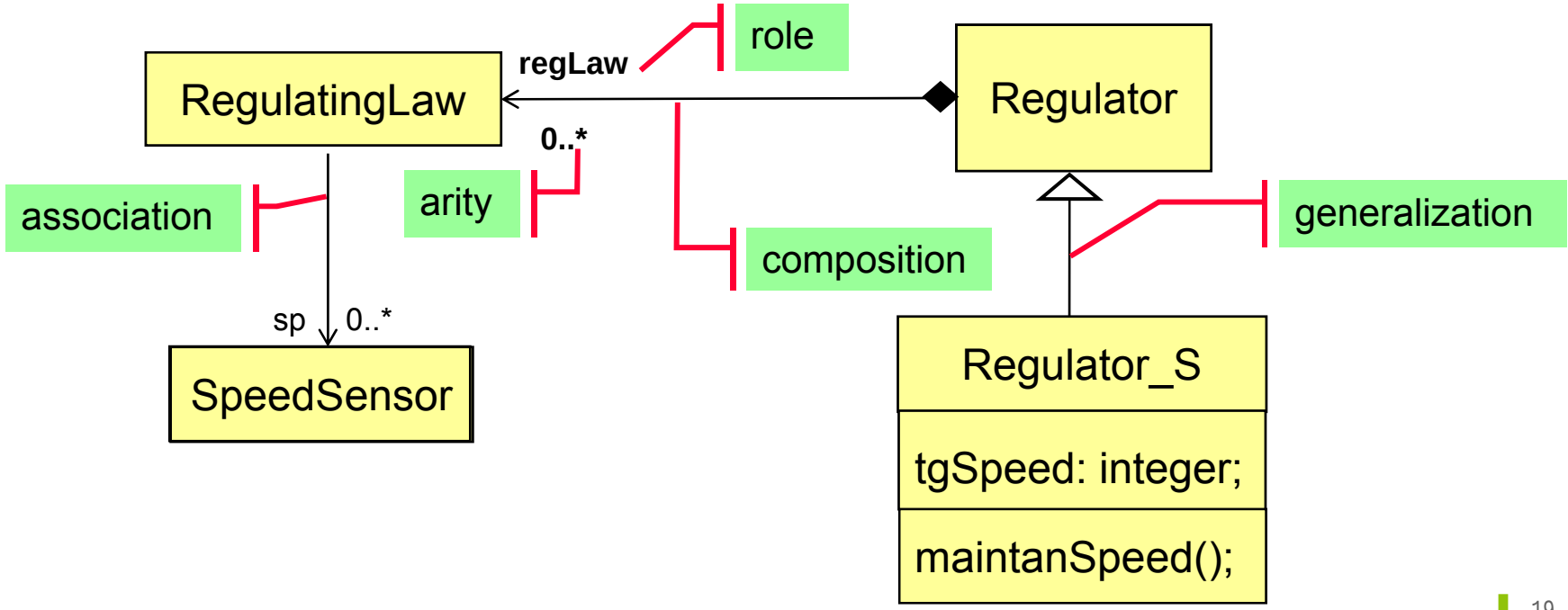
It denotes a semantic (or functional) usage between two classes (technical dependency).

Example

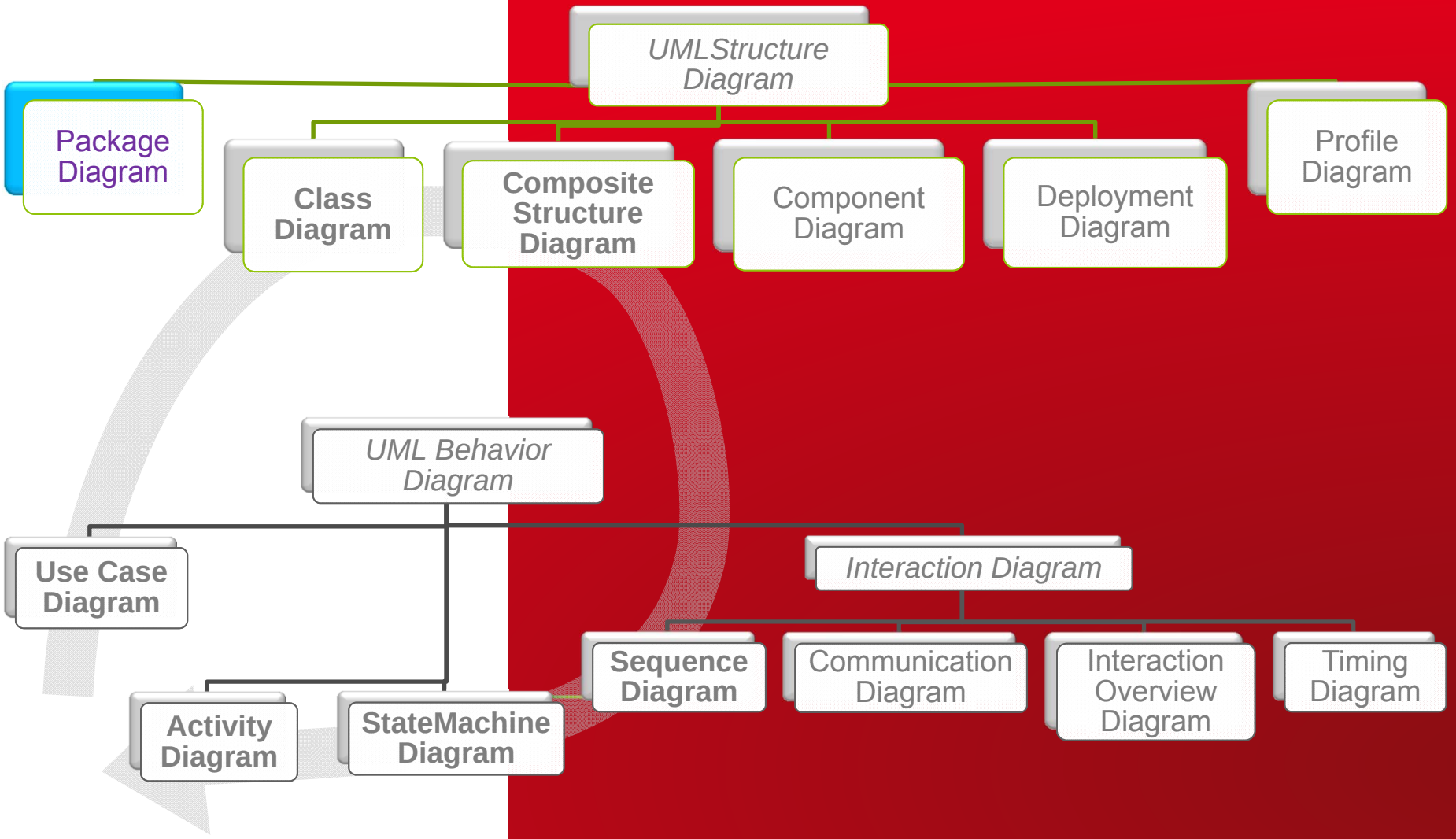


It introduces inheritance relationship

- link parent to children classes
- inheritance of features
  - Structural – Attributes & Relationships
  - Behavioral – Statemachine & Operations
- Multiple inheritance is possible

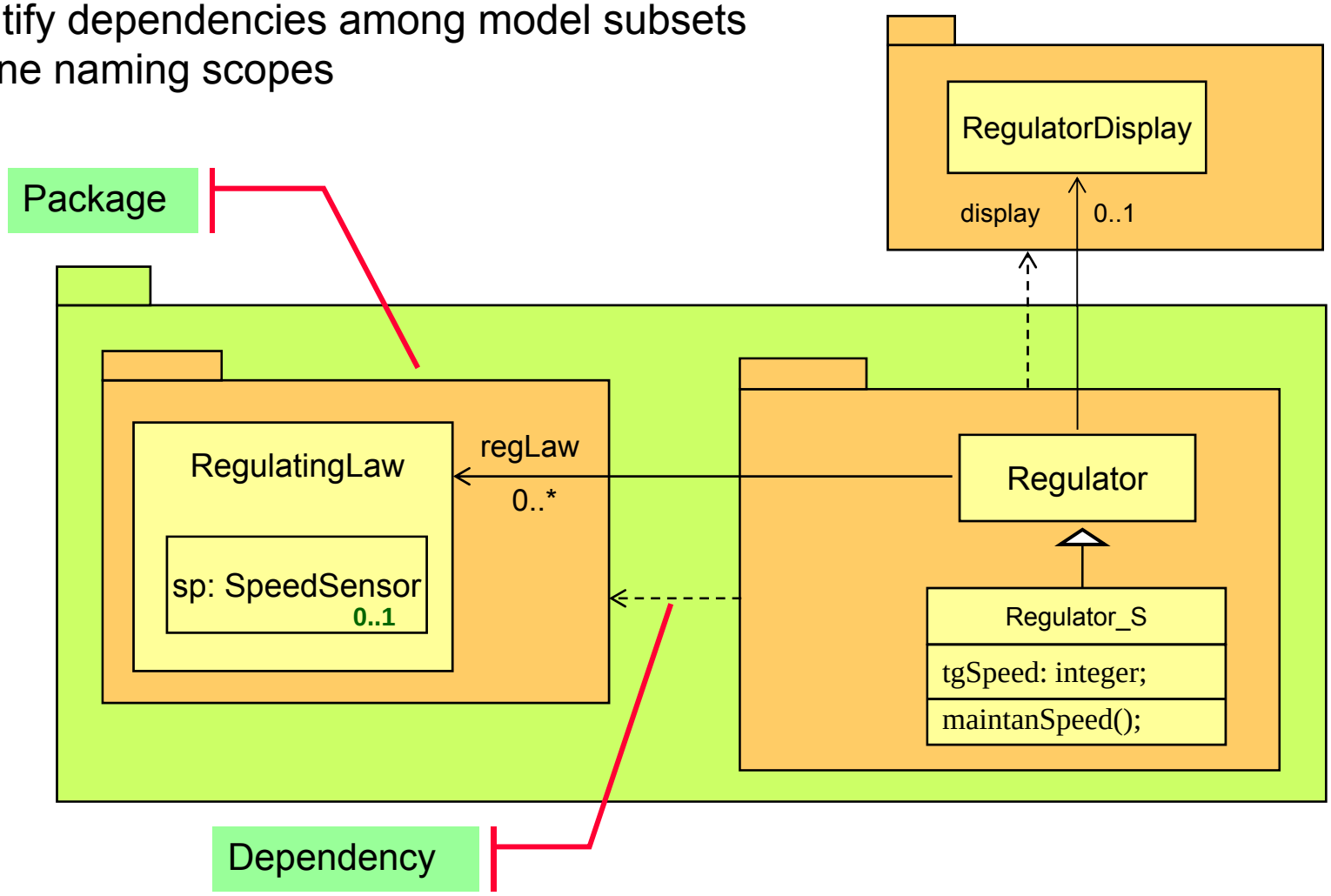


UML Package Diagrams



Just a way to:

- organize the models
- Identify dependencies among model subsets
- Define naming scopes



UML Communications

Communication: only by message passing

- **A message = an action + an event**
  - Usually point to point communication, but possibility to have a set of targets

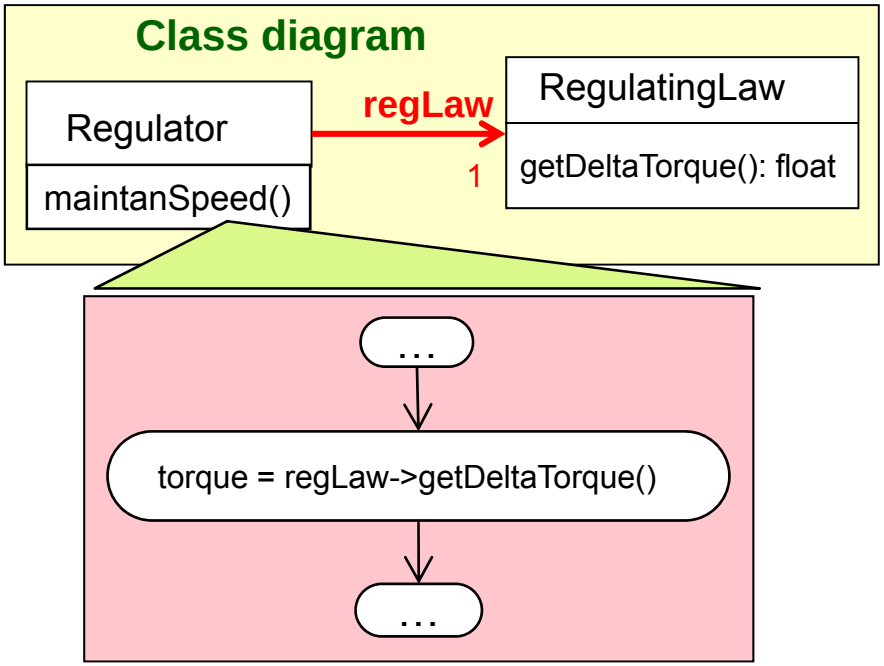
Two types of message sending

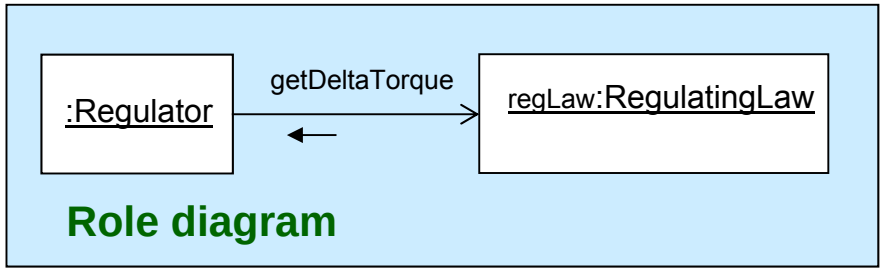
- **Operation call (CallAction + CallEvent)**
  - Synchronous/asynchronous, input and output parameters
    - Synchronous = wait the end of the process trigered by the call before to continue the caller process ➔ serialization
    - Asynchronous = continue the caller process as soon as the message has been sent (the call is made) ➔ parallelism

- **Signal sending (SendAction + SignalEvent)**
  - Asynchronous communication, input parameters only

In operation call:

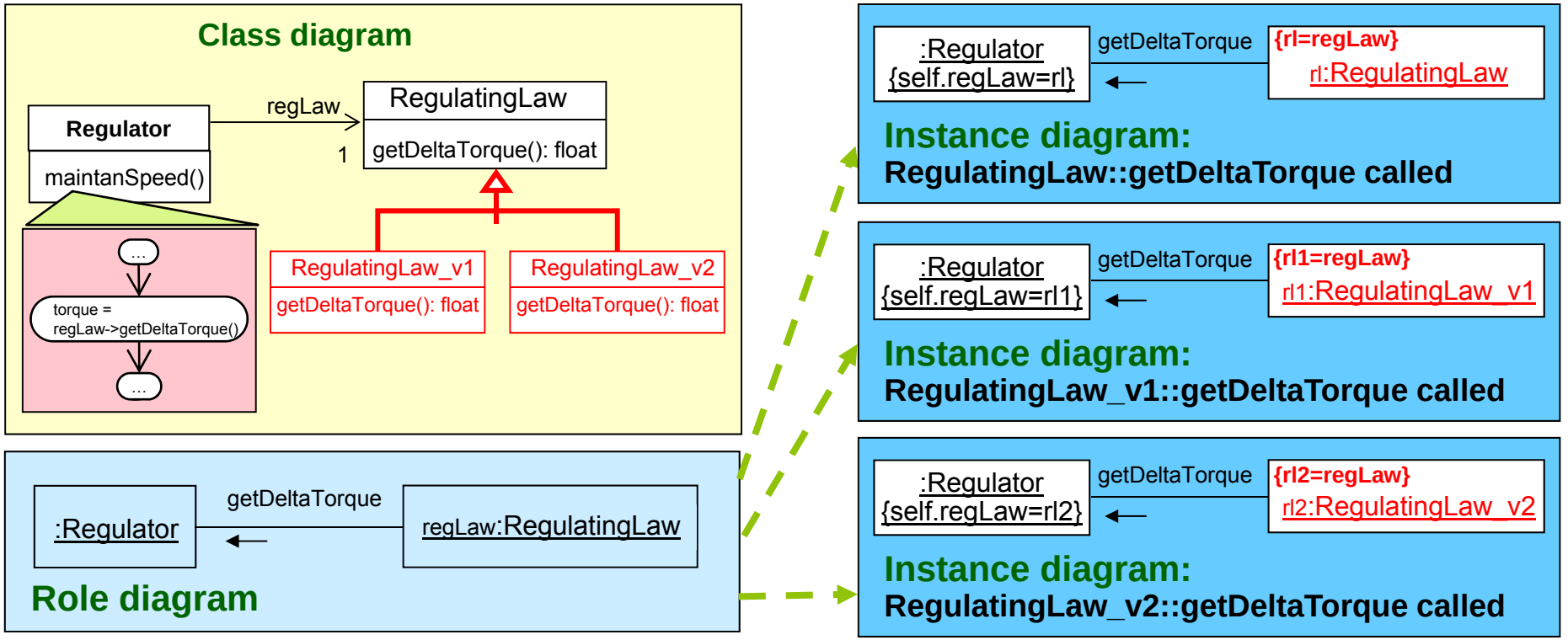
- Call requires explicit link (knowledge) between the sender and the receiver (target of the call)





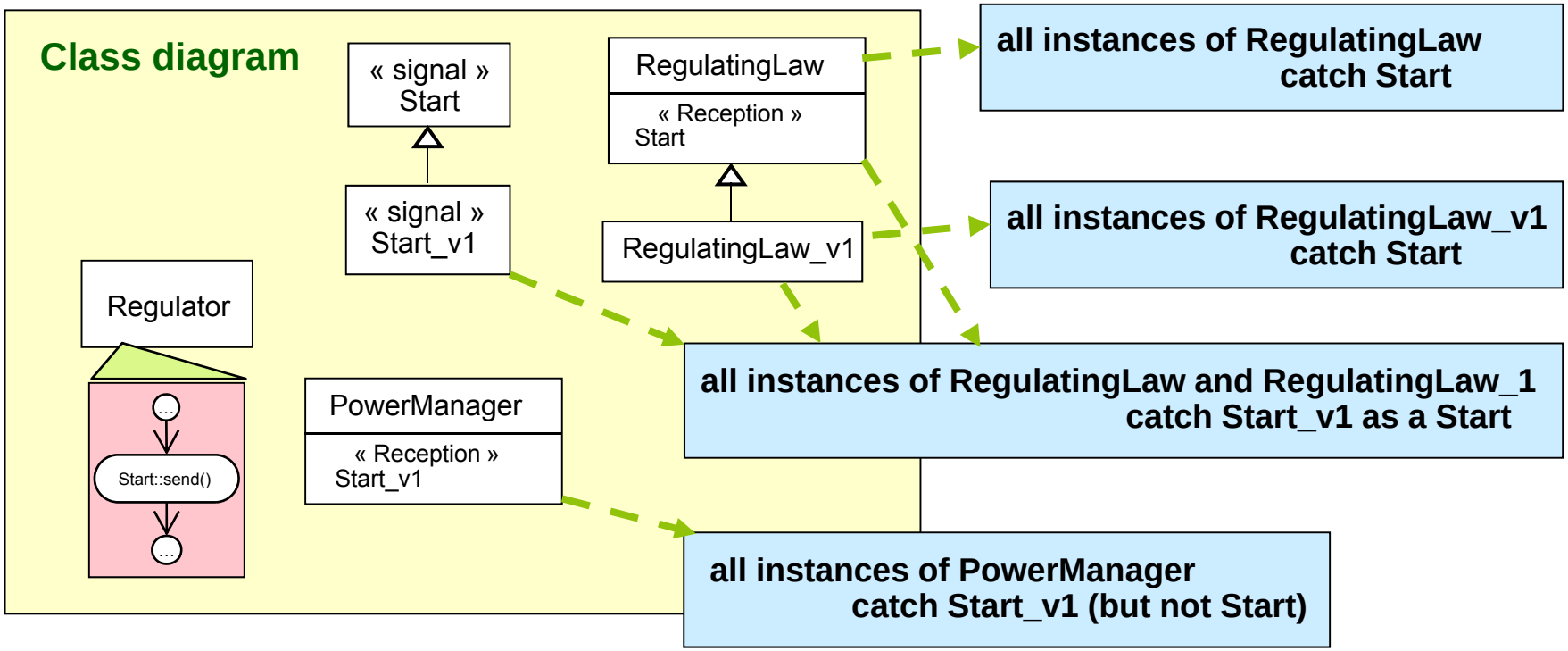
In operation call:

- Choice (static/dynamic) of called operation
  as defined by the actual target type



SIGNAL SENDING

- Independent declaration of the signals
- Possibly no explicit link required between sender and receiver
- Choice of target depending of target set definition
- Any inheriting signal maps to base class reception



Diapositive 26

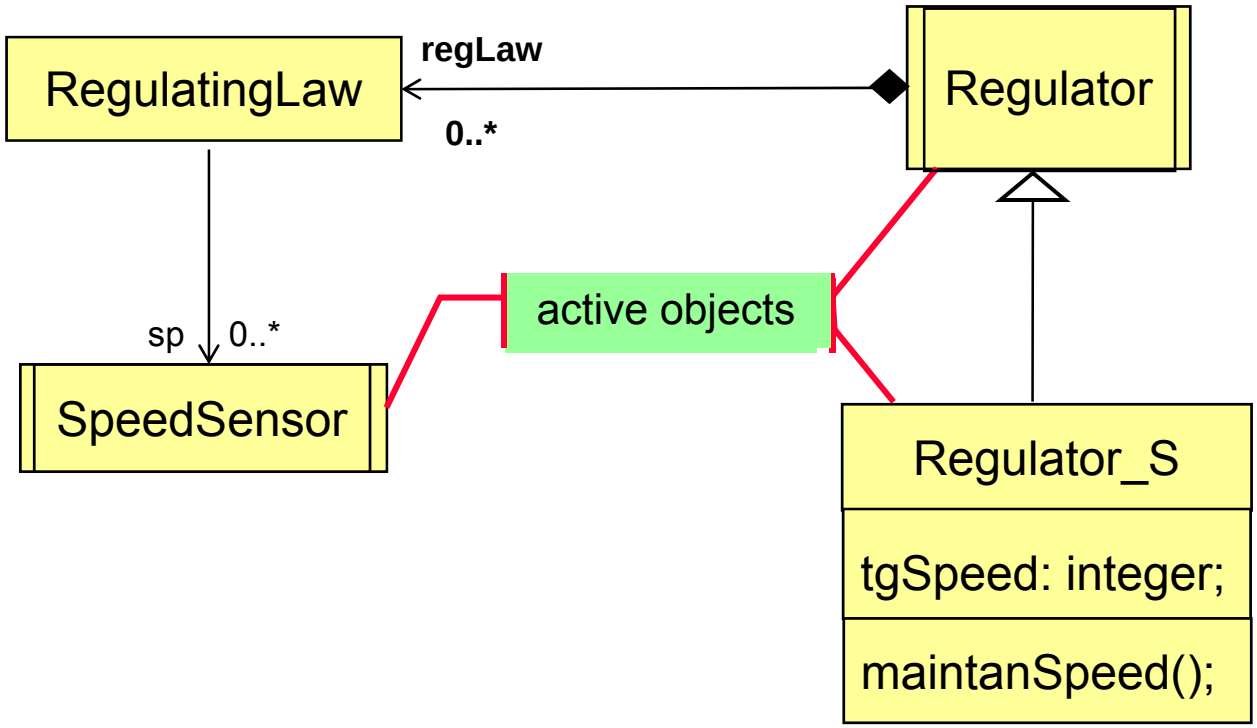
ft4 Clarifier l’émission

ft121910; 05/01/2006

UML Active Objects

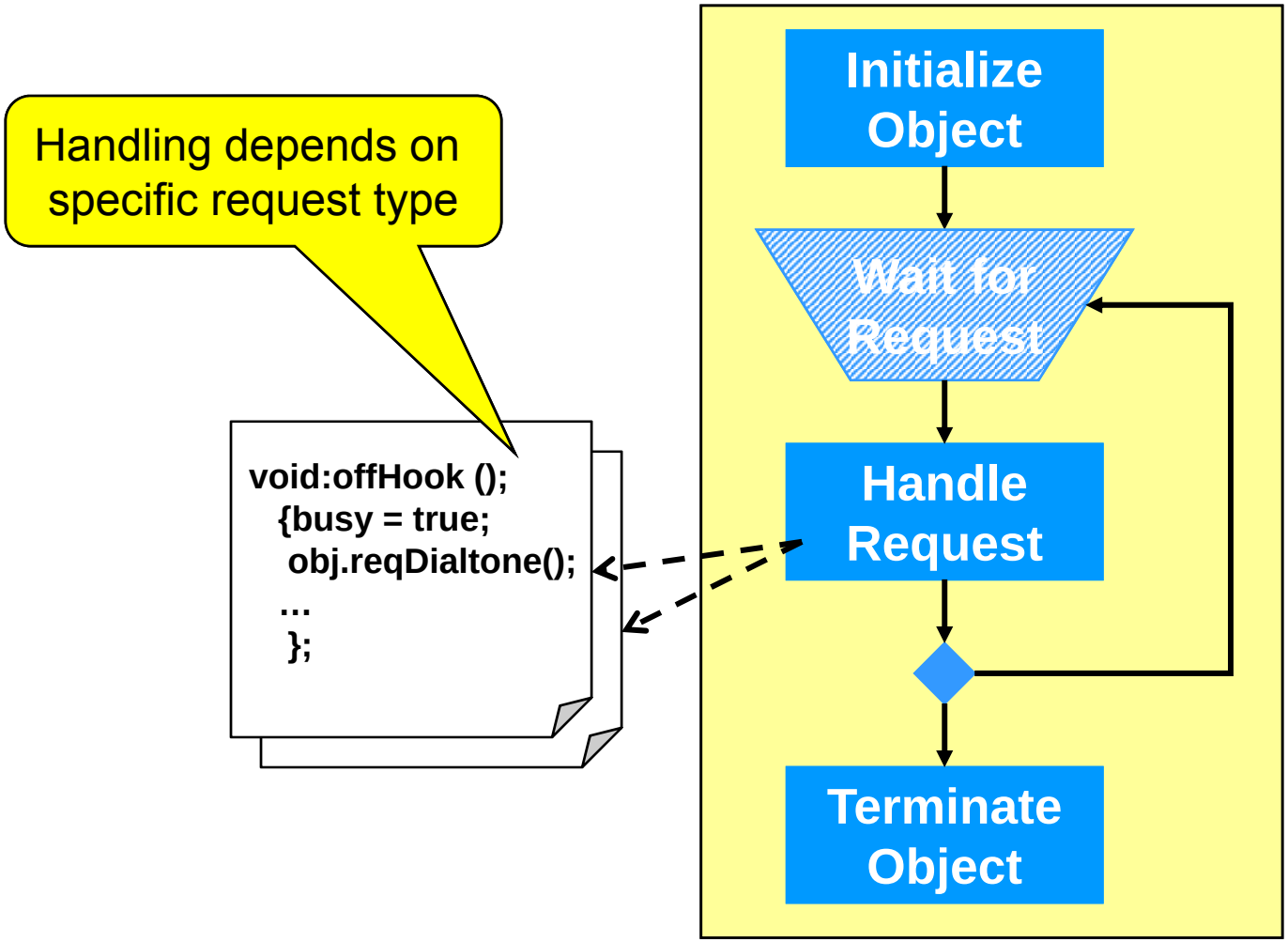
ACTIVE OBJECTS

→ Object types having their own execution thread(s)
→ A way to declare // entities inside the model
→ Implementation agnostic
(virtual, task based, multi-cpu, distributed systems, etc.)



Simple server model:

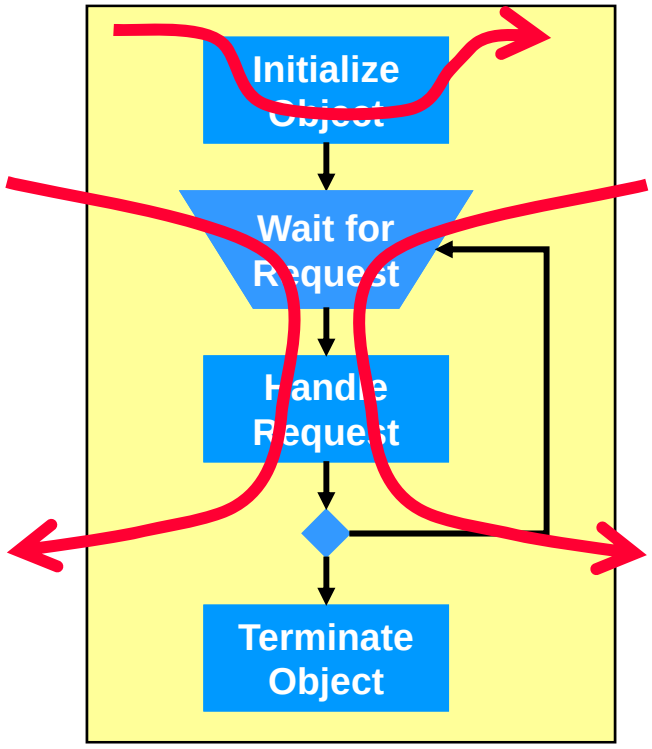
Passive or active objects



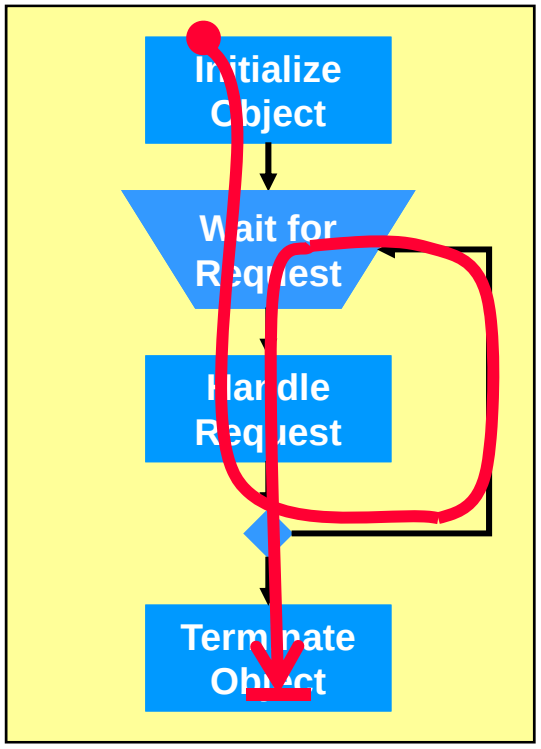
Passive objects: depend on external active resource (e.g. thread of execution…)

Active objects: self-powered (e.g. own their thread of execution)

Passive object

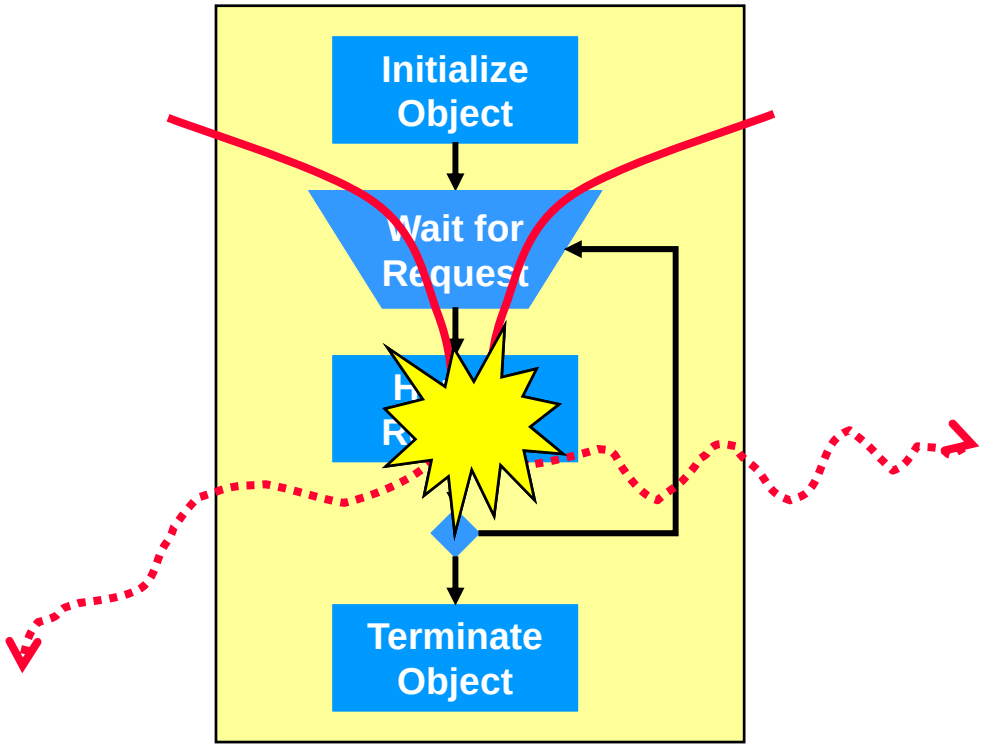


Active object



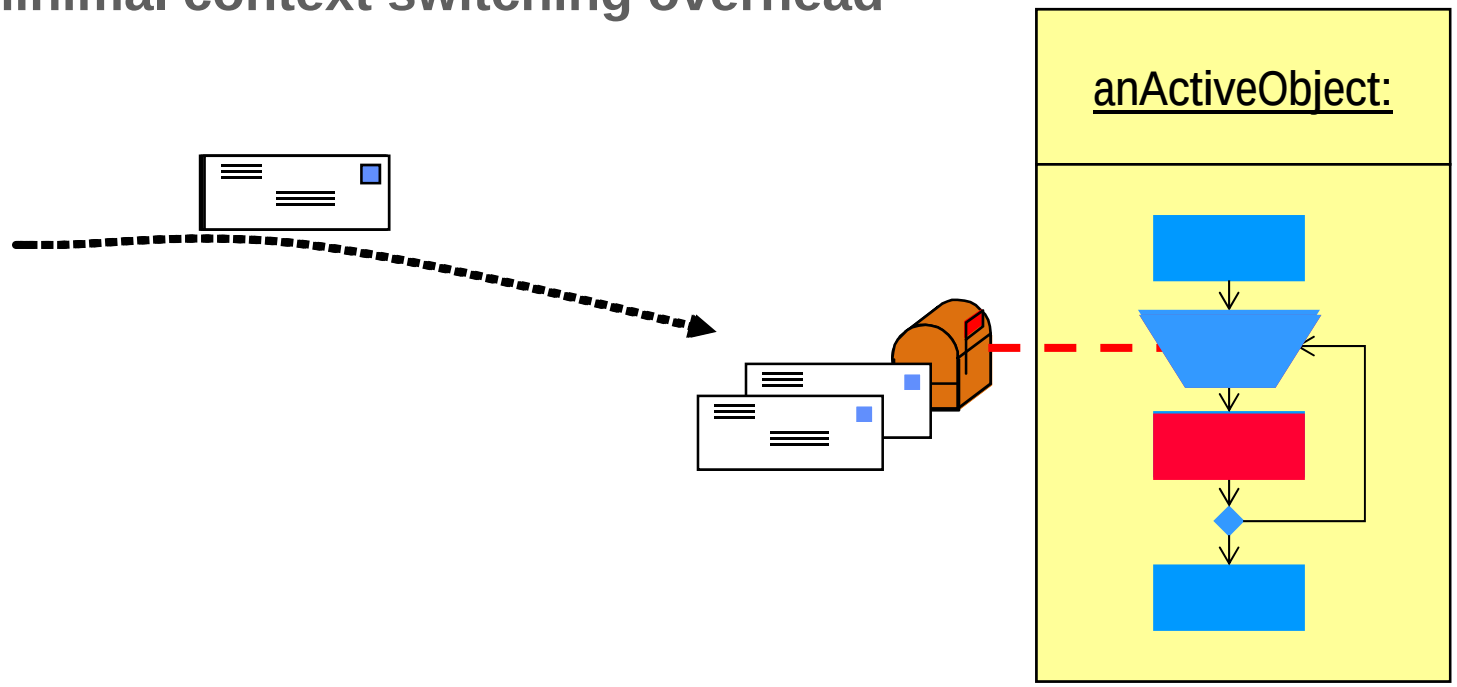
Encapsulation does not protect the object from concurrency conflicts!

Explicit synchronization is still required

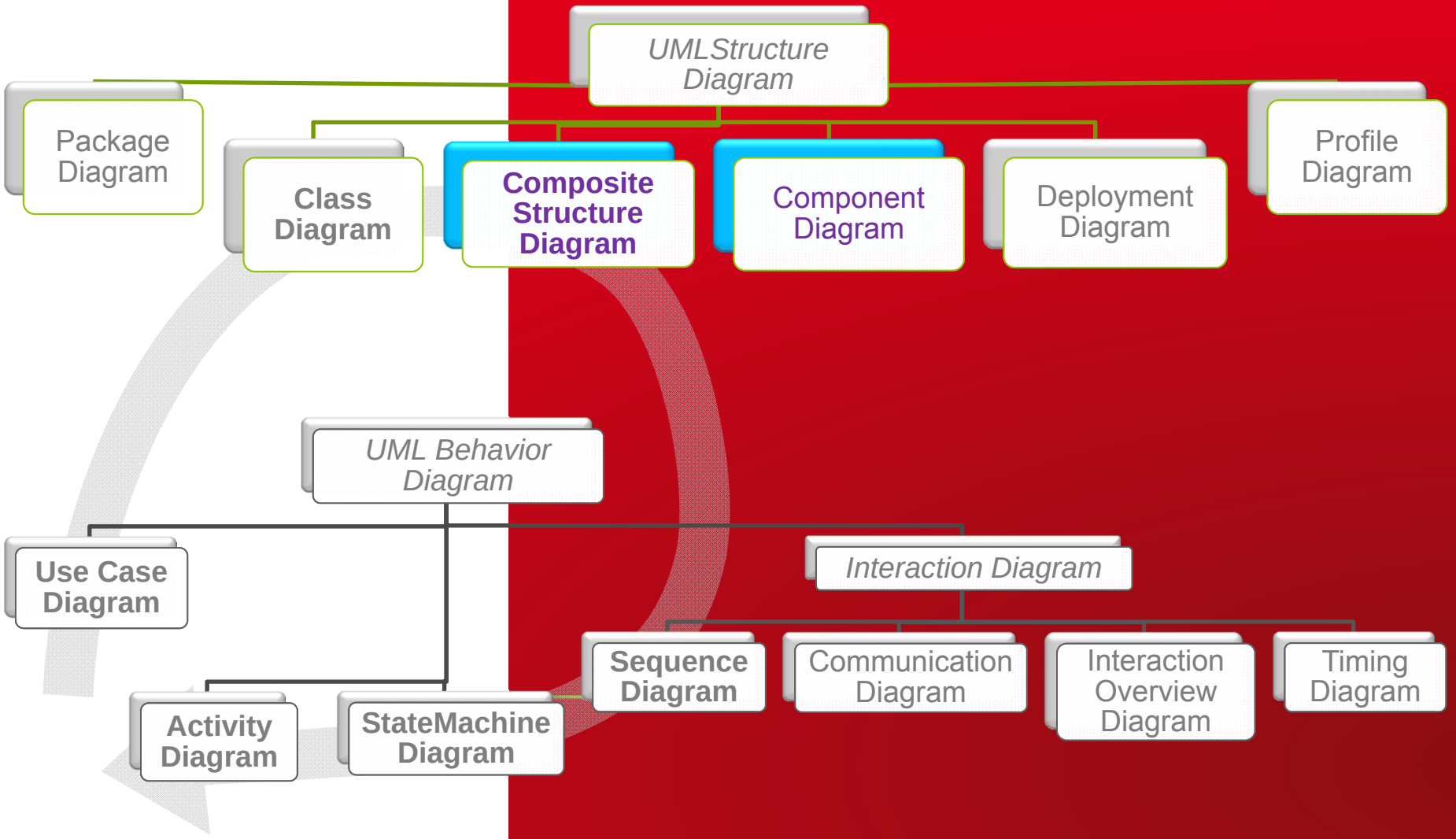


Run-to-completion model:

- serialized event handling
- eliminates internal concurrency
- minimal context switching overhead



UML Compoent Diagrams



NOTION OF COMPONENT: TWO VIEWS

An external view (or “black-box” view)

- Publicly visible properties and operations.
- Behavior may be attached to interface and to the component itself.
- Component wiring via **dependencies** between interfaces.

An internal view (or “white-box” view)

- Private properties and realizing classifiers.
- External and internal views mapping:
  - **Delegation** connectors to internal parts
  - More detailed behavior specifications (e.g. interactions and activities) may be used to detail the mapping.

OUTLINES OF THE COMPONENT CONCEPT

Self contained unit that encapsulates the state and behavior of a number of classifiers by specifying:

- **Interfaces**
  - Provided interfaces
    - Formal contract of the services available for clients.
  - Required interfaces
    - Requirements from other components or services in the system.
- **Or ports**
  - Typed by required or/and provided interfaces

Substitutable unit that can be replaced at design time or run-time by a component that offers equivalent functionality based on compatibility of its interfaces.

INTERFACES OF COMPONENTS

Required and provided interfaces allow for the specification of:

- **Structural features (attribute, association…)**
- **Behavioral features (operation/reception, statemachine…)**

Provided interfaces may be directly implemented by a component or by some of its realizing classifiers.

Required interfaces are designated by a usage dependency from the Component or one of its realizing classifiers.

Required and provided interfaces may optionally be organized through ports.

OUTLINES OF THE INTERFACE CONCEPT

Declaration of a set of coherent public features and obligations.

Contract that any instance realizing it must fulfill.

- **Possible additional constraints**
  - pre- and post-conditions
  - protocol state machine that imposes ordering restrictions on interactions through one interface.

Since interfaces are declarations, they are not instantiable.

- **Either realized by a component or parts of a component.**
- **Or realized by ports attached to component (or a composite class).**

A given classifier may implement more than one interface.

One interface may be implemented by a number of different classifiers.

NOTATION FOR EXTERNAL VIEW ("BLACK-BOX" VIEW)

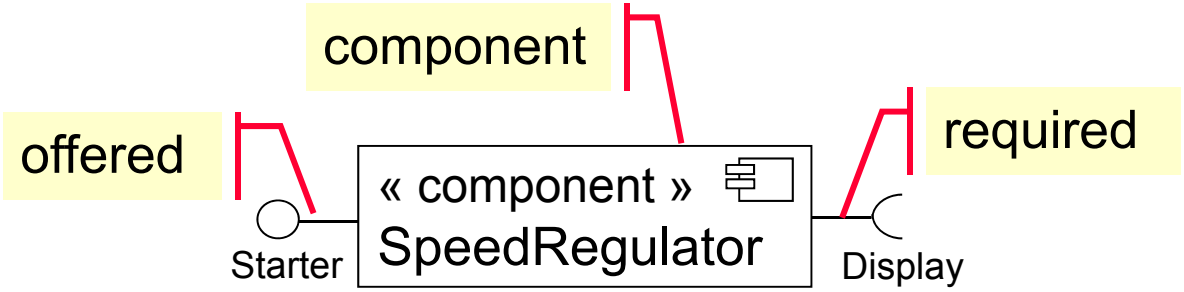


Figure1: condensed notation

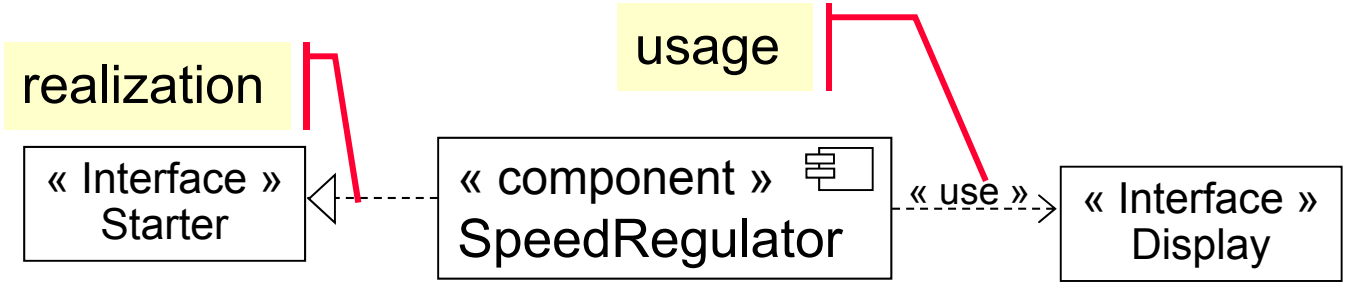


Figure2: notation with explicit interfaces

➔ PUBLIC, PARTIAL AND ABSTRACT VIEW OF A CLASSIFIER

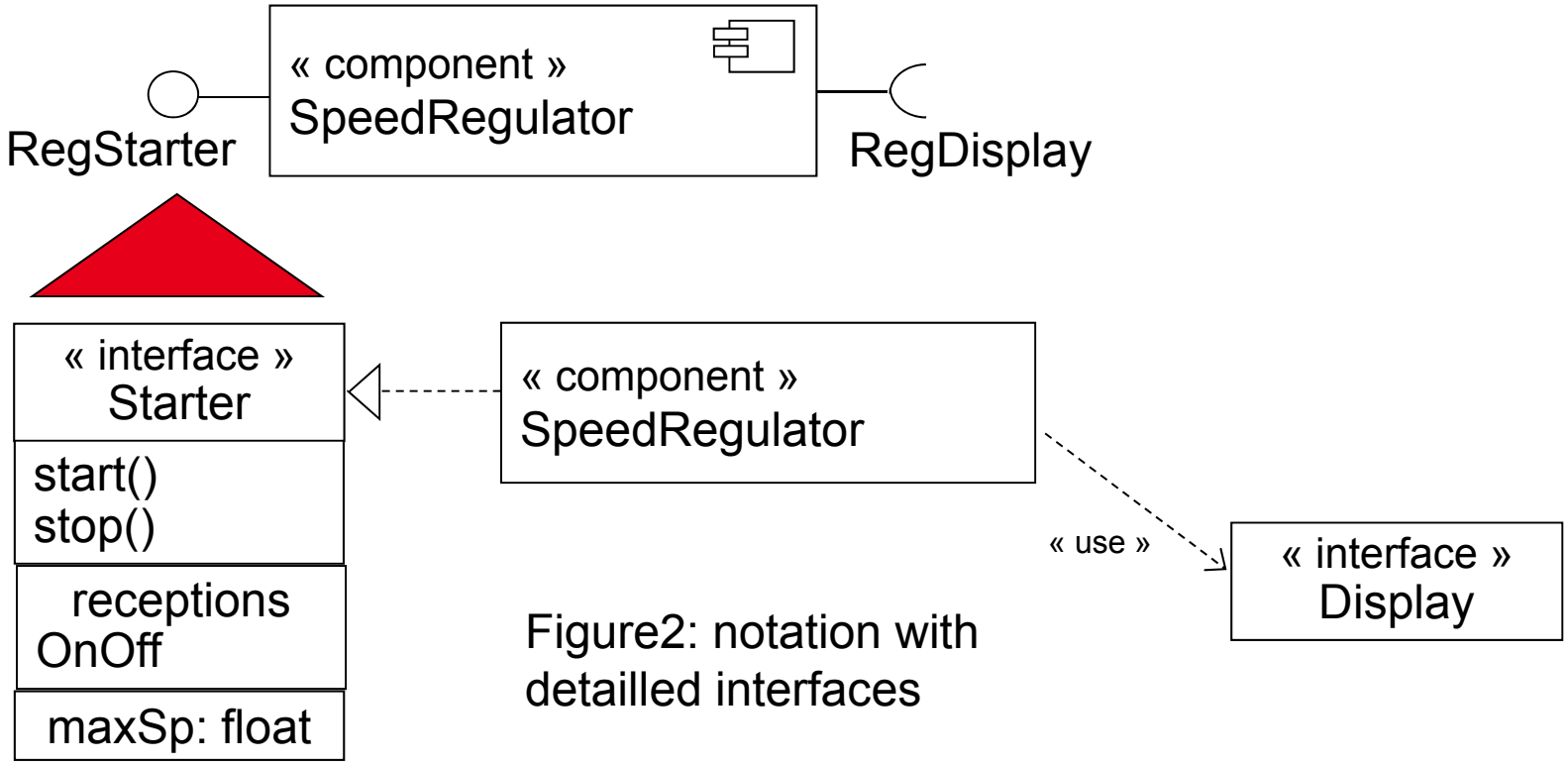
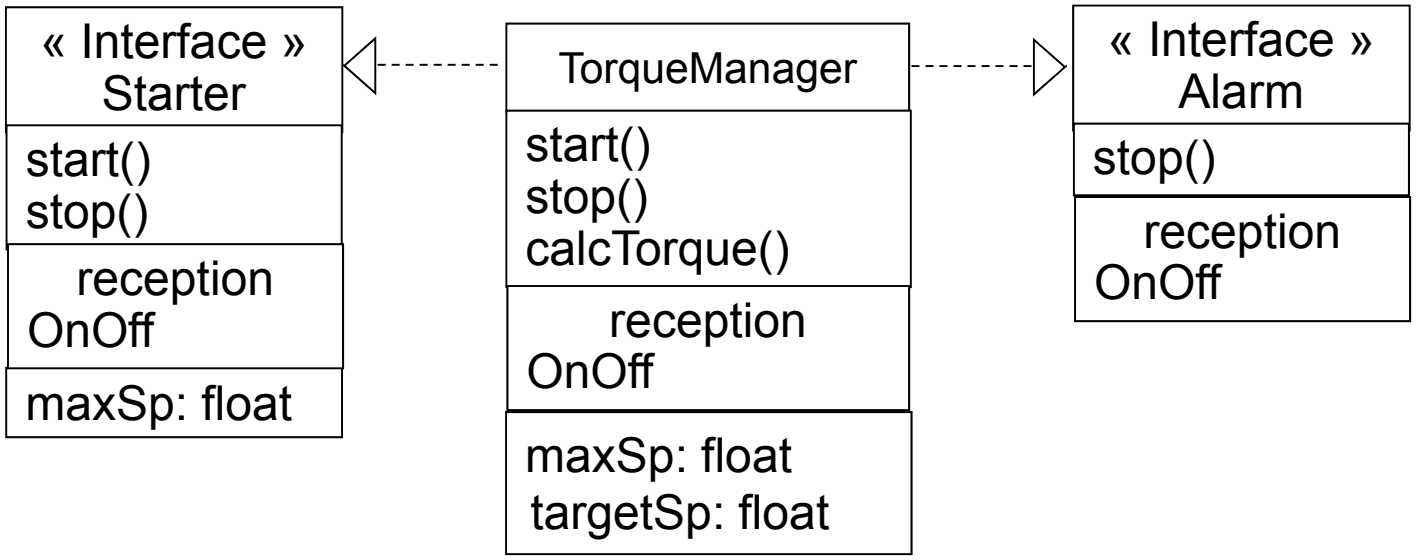


Figure2: notation with detailled interfaces

➔ **Defined as types, apply to classifiers and component types**

INTERFACE: REALIZATION



- **Realization relationship is a conforming realization dependencies**

www.eclipse.org/papyrus

Commissariat à l'énergie atomique et aux énergies alternatives
Institut Carnot CEA LIST
Centre de Saclay | 91191 Gif-sur-Yvette Cedex
T. +33 (0)169 077 093 | M. +33 (0)688 200 047

Direction DRT
Département DILS
Laboratoire LISE

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019