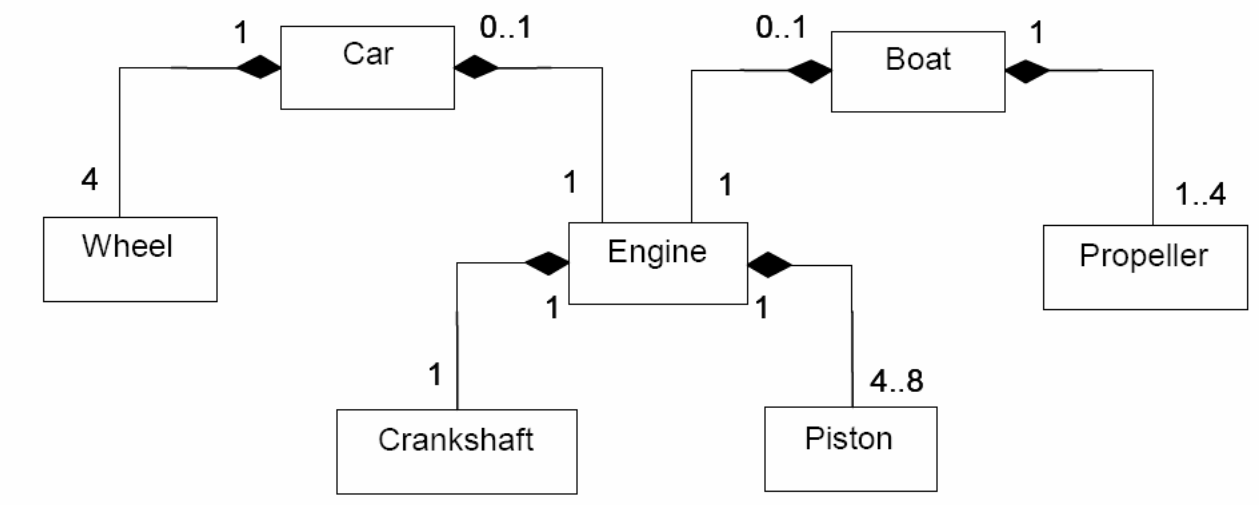
Composite Structures

UML 2 Composition Model

- Purpose: improve the “black diamond” composition
- Supports connections between parts at the same level of decomposition, in addition to the usual part-whole associations.
- Complex networks of entities can be represented within a single class, inherited to subclasses, with links maintained between objects playing parts at runtime.
- StructuredClassifiers
- See:
  - UML 2.0 Superstructure
  - C. Bock, “UML 2 Composition model”, JOT, vol 3, n° 10, 2004, pp 47-73

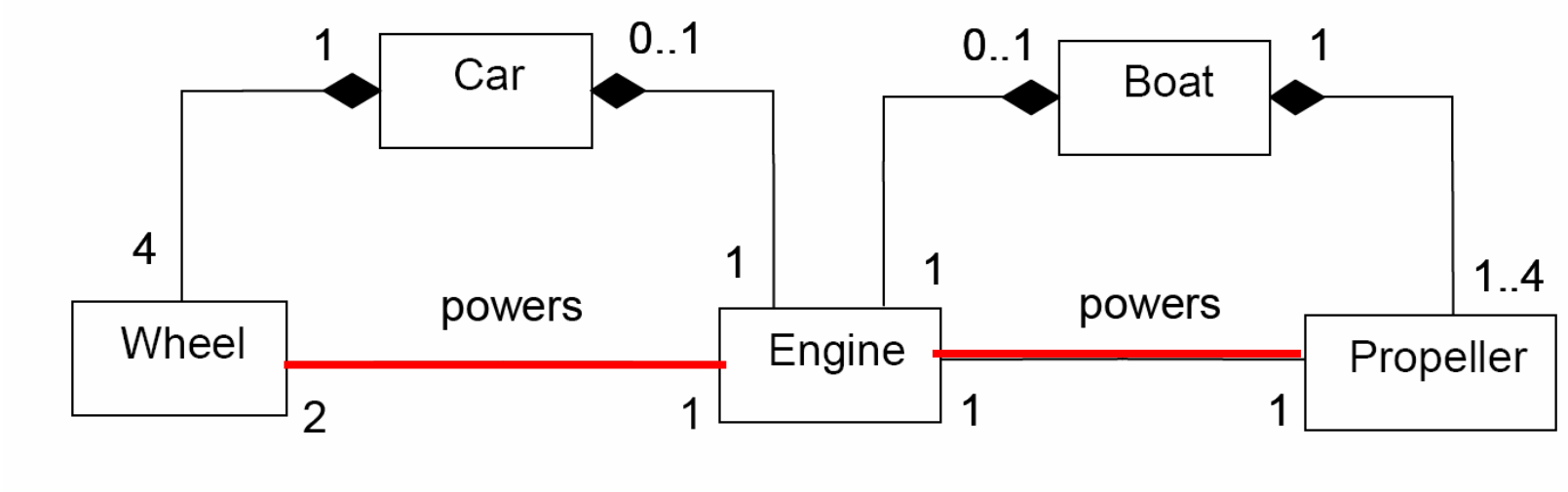
UML 1.x Composition



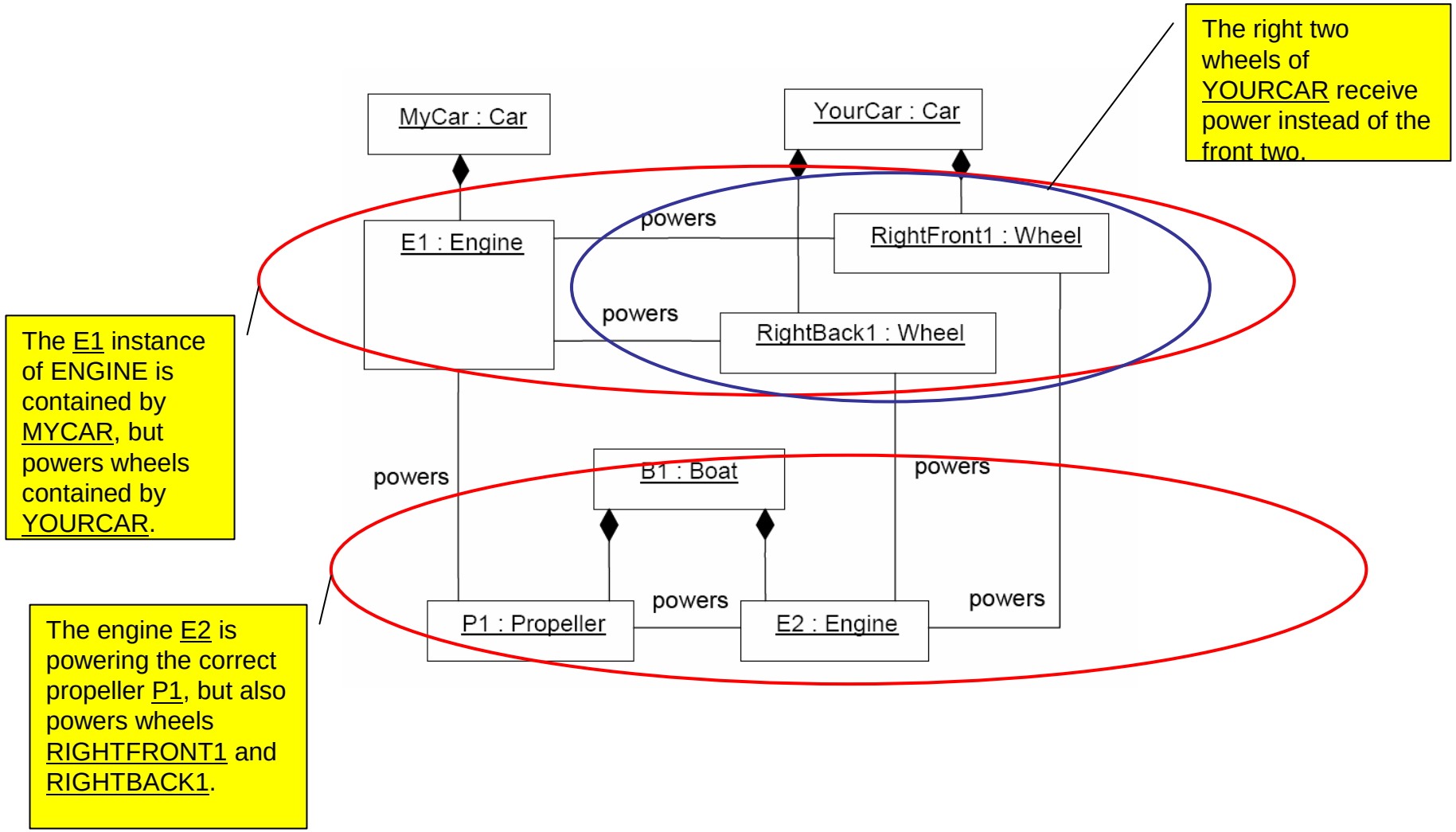
- UML 1.x composition is familiar to users as the black diamond notation on associations (called *composite aggregation*, or informally “strong composition”).
- The black diamonds mean the same instance of ENGINE cannot be used simultaneously in an instance of CAR and an instance of BOAT, even though the class ENGINE is shared between the classes CAR and BOAT.
- The black diamond also means that an instance of ENGINE is destroyed when its containing instance of CAR or BOAT is destroyed.
- Note: Classes in the physical examples only describe information records of real or imagined physical objects.

Associations at the same level of decomposition

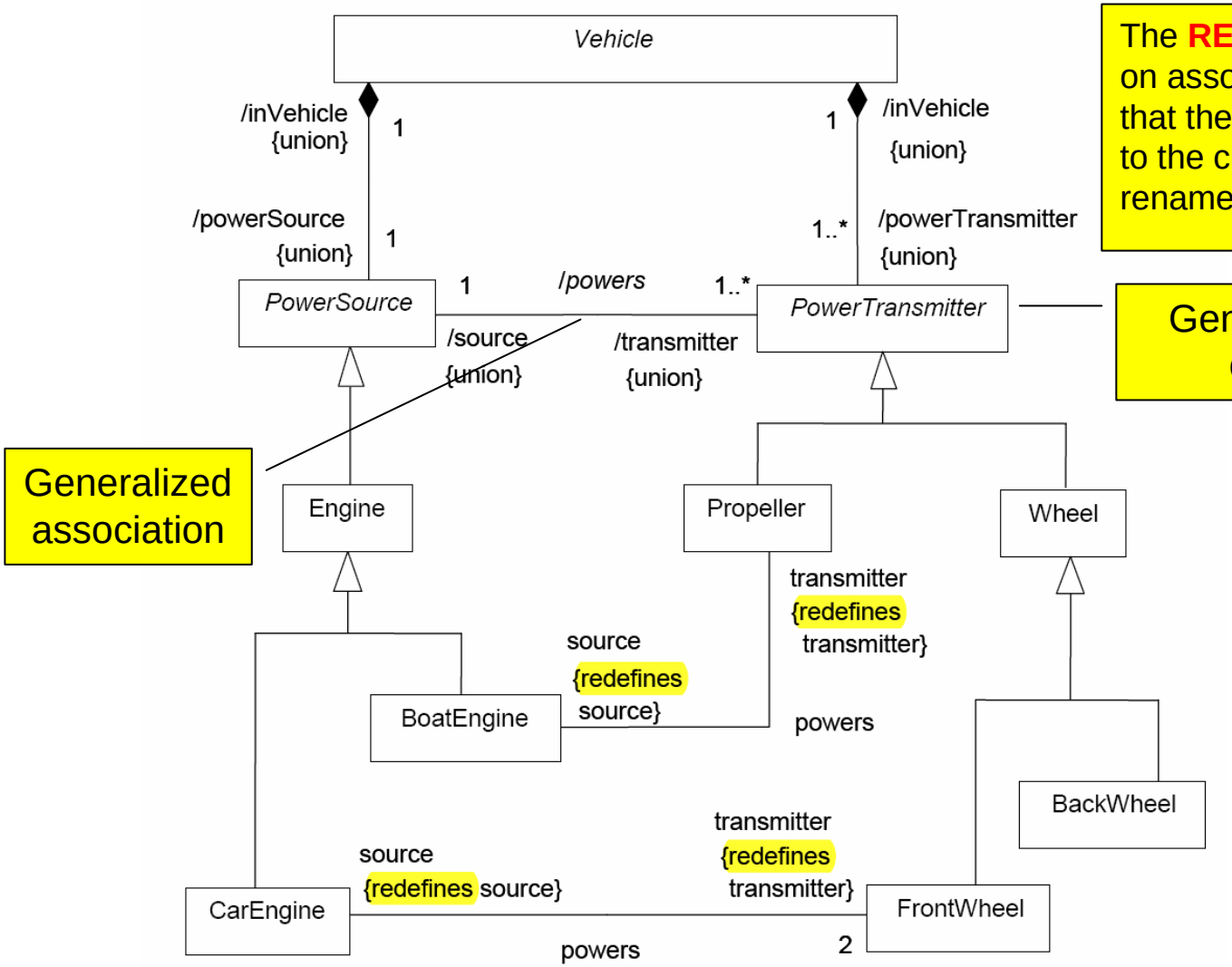
- While the UML 1.x model is fine for hierarchical decomposition, as in the figure in the previous slide, it has significant limitations when connecting elements at the same level of decomposition.
- The associations in the figure below are defined globally for all engines and boats, not in the context of individual cars and boats. This leads to erroneous interpretations.



An incorrect instance model



Using generalized classes



The **REDEFINES** property on association ends indicates that the association is restricted to the class at that end, and renamed for that purpose.

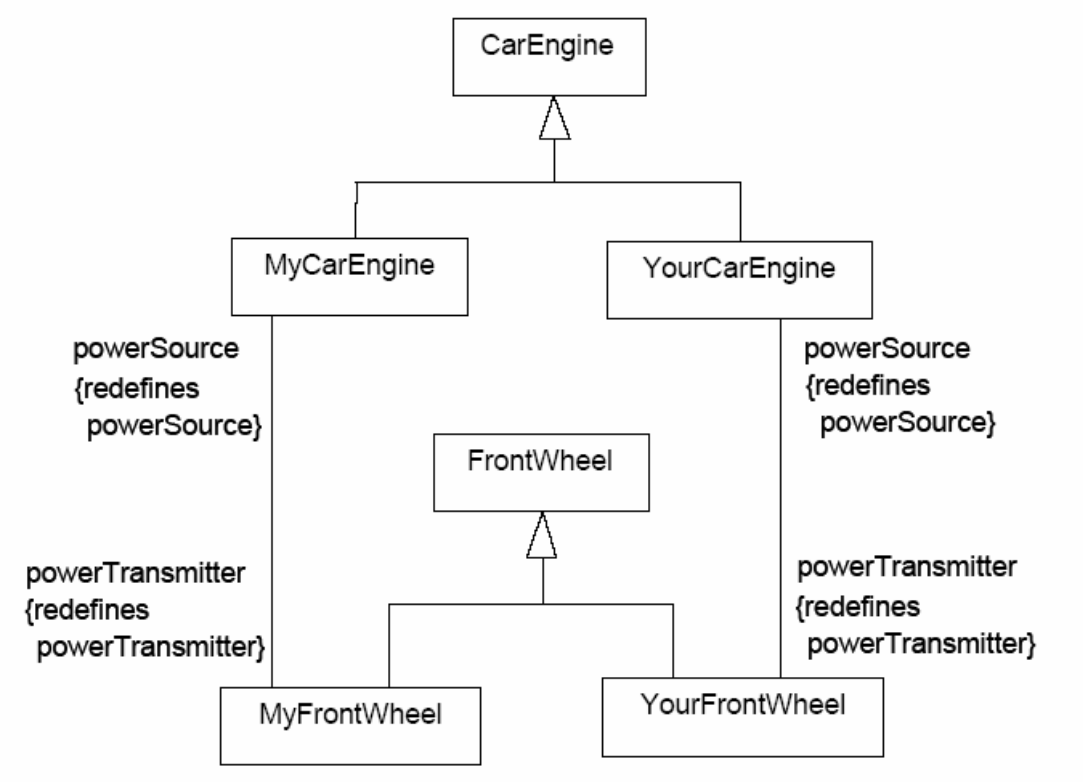
This diagram is borrowed from C. Bock's paper.

But it is not fully correct.

See a better solution later.

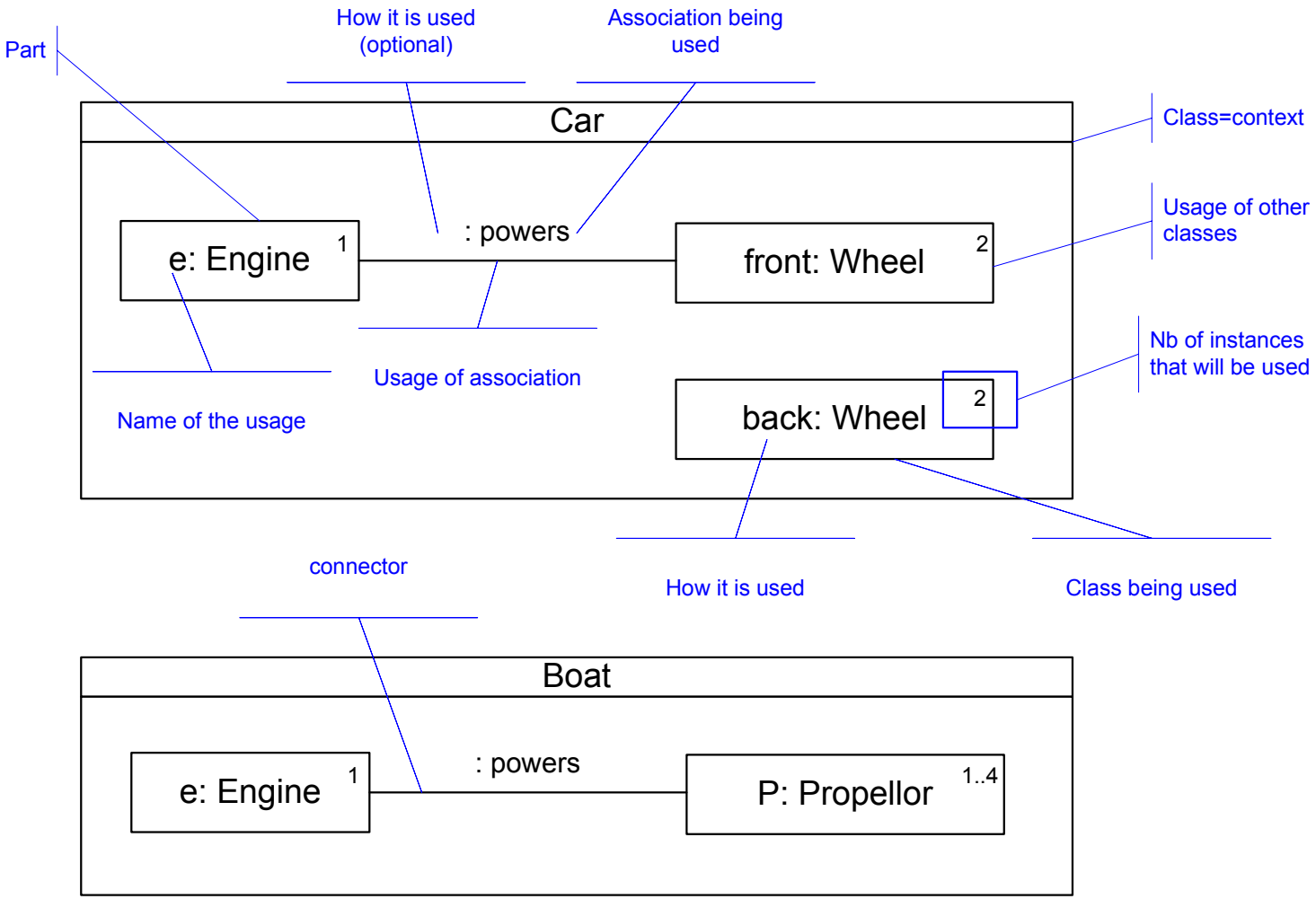
And yet, still misinterpretations

However, even the above cumbersome class model do not prevent an engine in one car from powering wheels in another. This requires each individual car to have classes for its particular wheels and engine, so the association can be further specialized in each car. This is obviously impractical.



UML 2 Composition (1)

New diagram specifically for **composite structure**



Connector

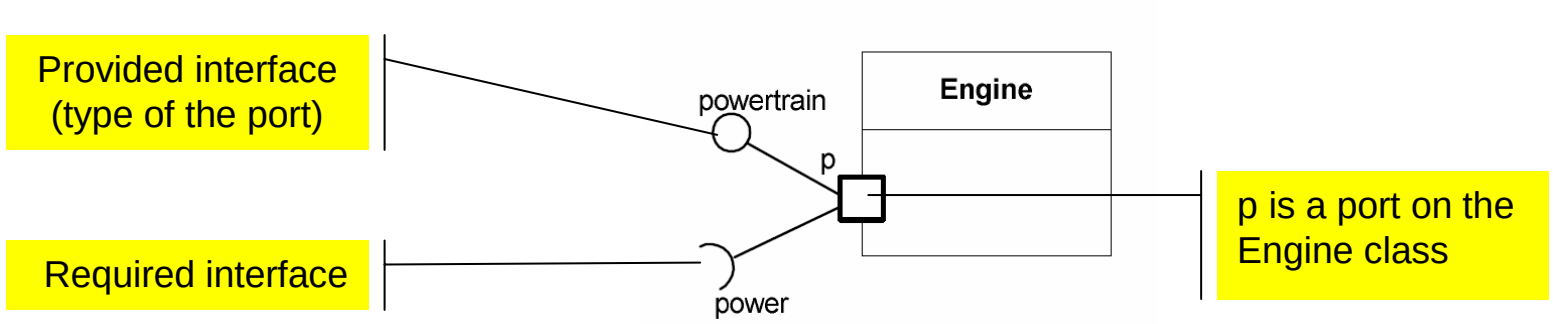
- Connectors represent communication links between instances of classes participating in an internal structure.
- They can be runtime instances of associations, or they can represent dynamic communication set up at runtime.
- Note that while associations between classes represent a link between any instances of those classes, a connector represents a link between only the two instances specified at each end of the connector (Definition vs. Usage)

Connector end types

- UML specifies several rules for determinating the types of the elements at each end of a connector:
  - If a connector represents an instance of an association, the types of the instances at either end of the connector must be the same types at either end of the association.
  - If an instance at one end of a connector has required interfaces, the instance at the other end of the connector must provide one of those interfaces.
  - If an instance at one end of a connector has required interfaces and a port is connected to the other end of the connector, the port must provide a required interface.

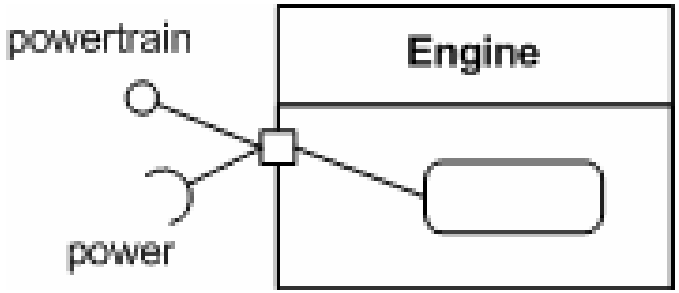
Ports (1)

- A port is a way to offer functionality from a composite structure without exposing the internal details of how the functionality is realized.
- Ports are associated with required/provided interfaces. Required interfaces show what the owning classifier may ask of its environment through a given port. Provided interfaces show what functionality a classifier exposes top the environment.



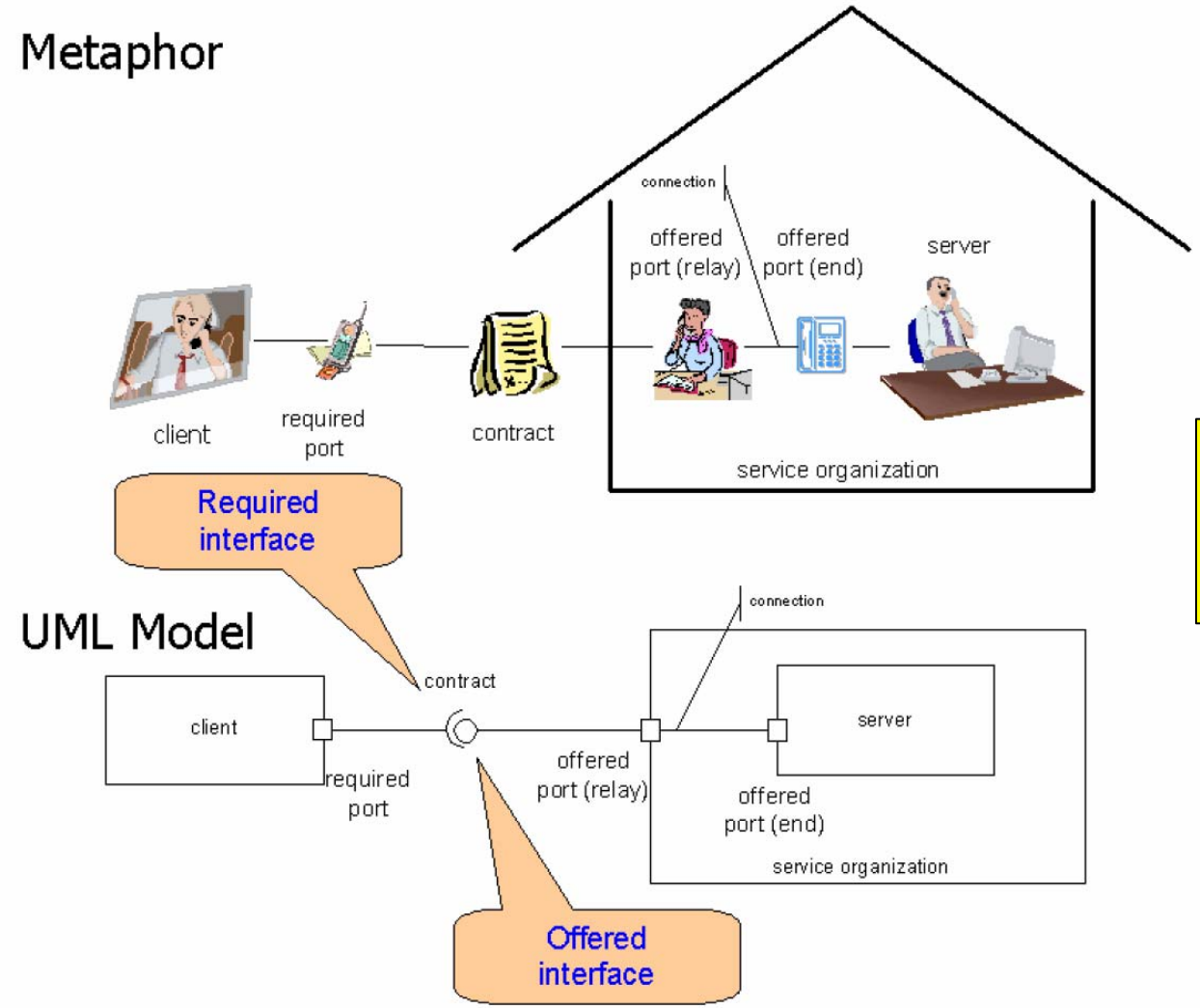
Ports (2)

- Behavioral port: the classifier owning the port provides the implementation of the functionality.



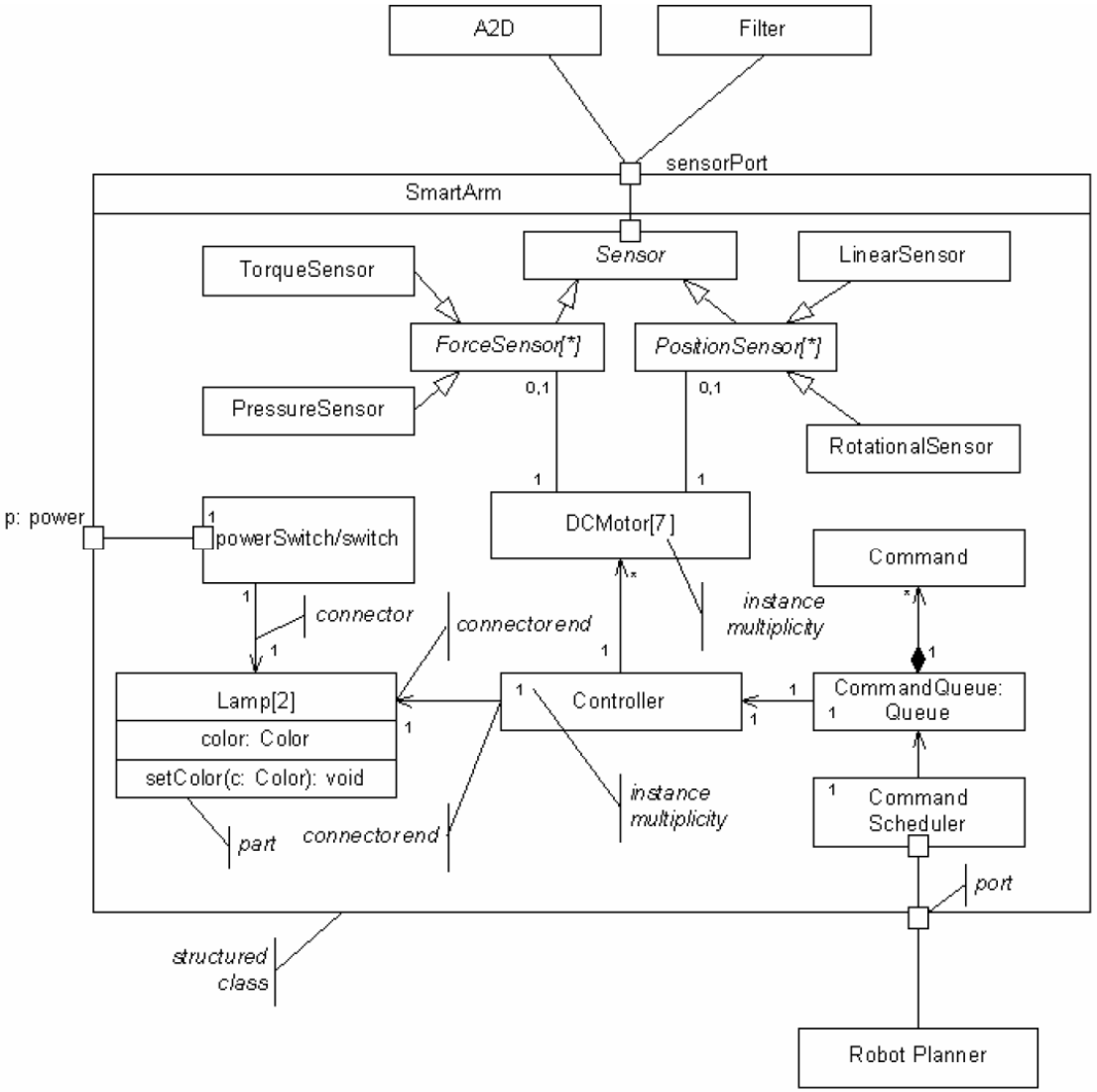
- Service port: the functionality is realized by internal elements.

Simple Example of Ports



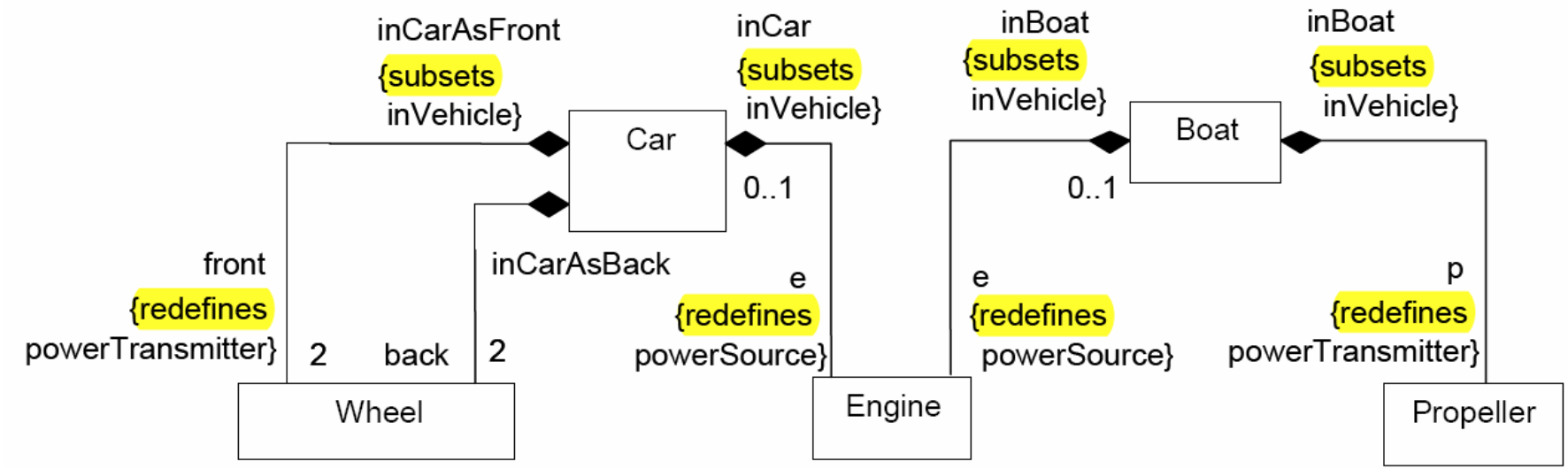
B.P. Douglass “UML 2.0: Incremental Improvements for Scalability and Architecture”, I-Logix 2003.

More elaborate example of Ports and Interfaces



UML 2 Composition (2)

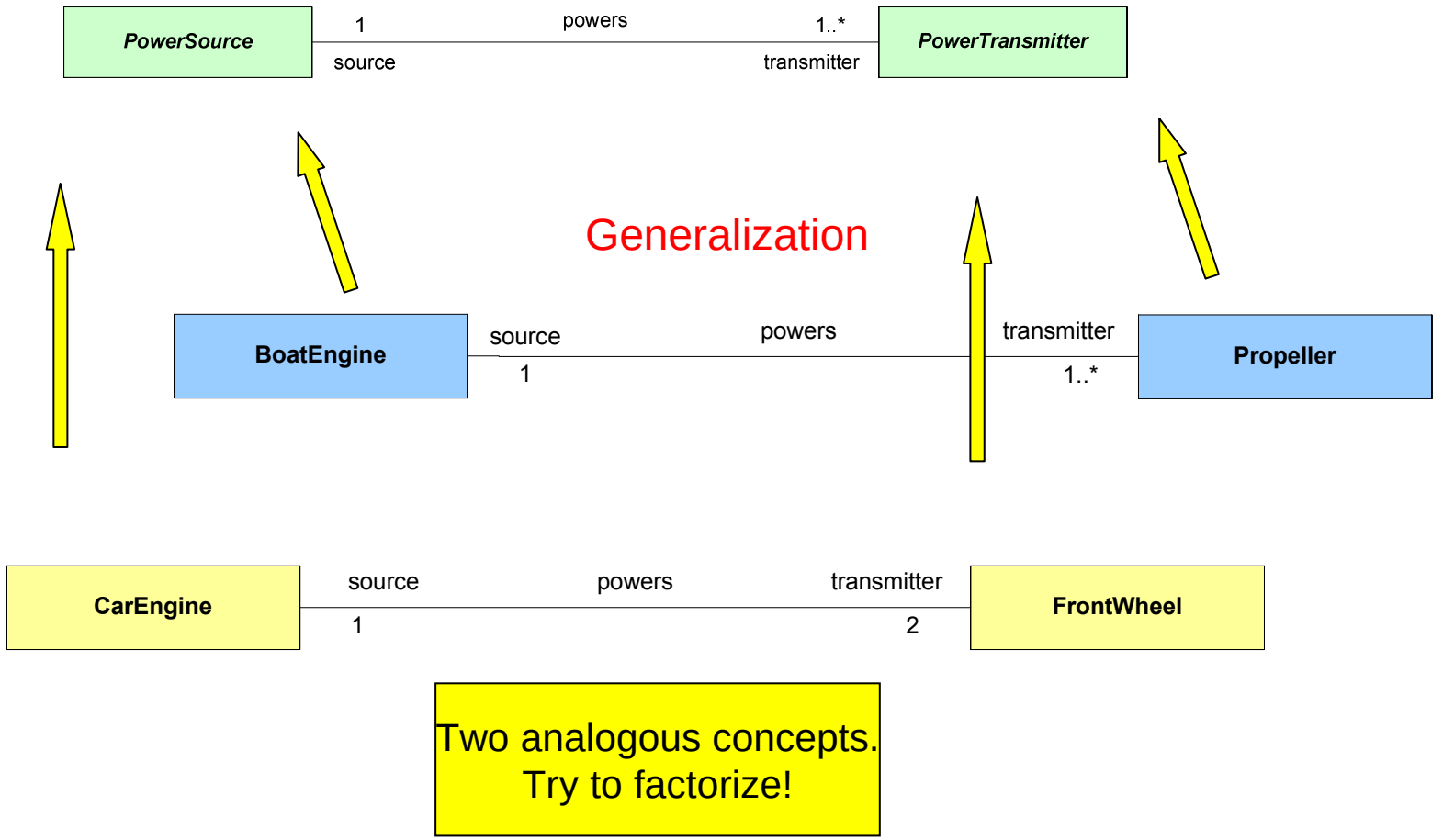
An alternate representation:



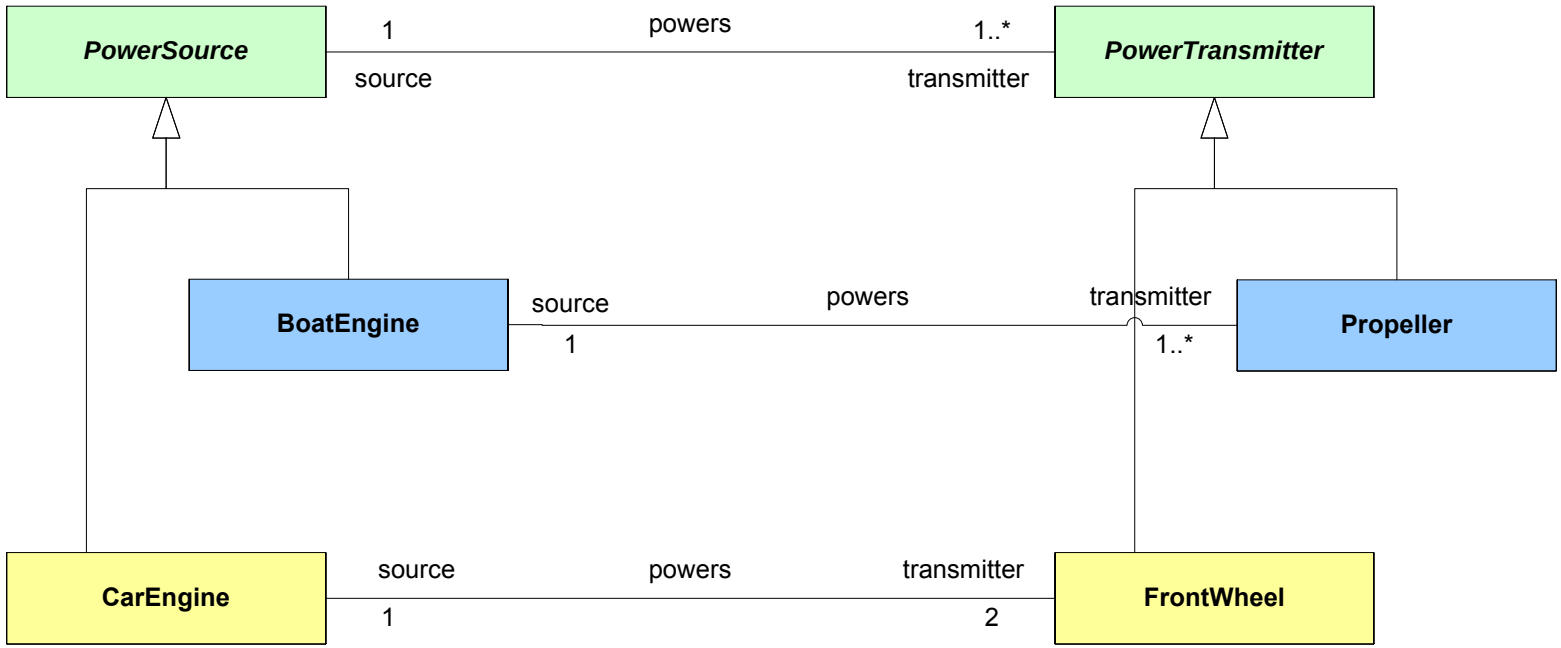
The **SUBSETS** property on association ends is another association specialization capability in UML 2. It indicates the association is specialized and renamed at that end, but does not restrict the inherited association to the class at that end (compare to REDEFINES).

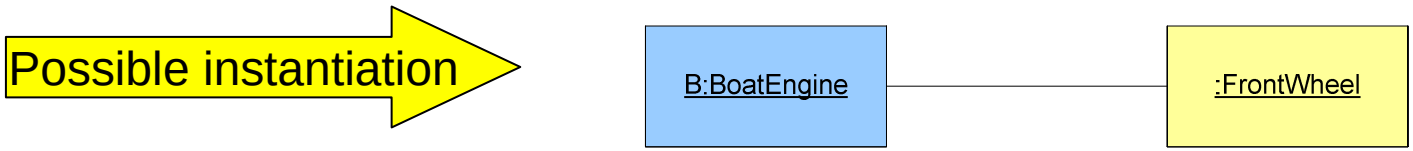
There is still a problem with redefines

Redefines/Subsets revisited (1)



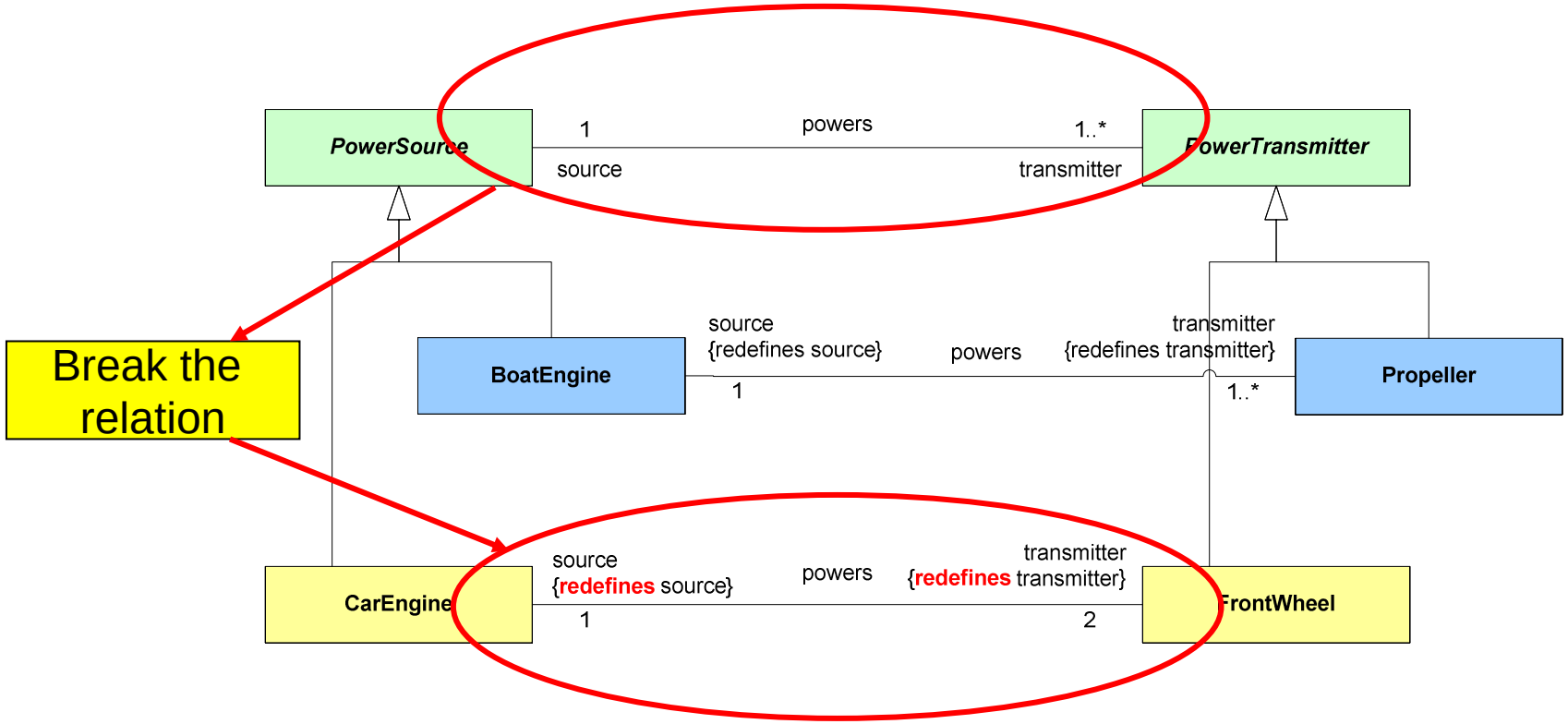
Redefines/Subsets revisited (2)



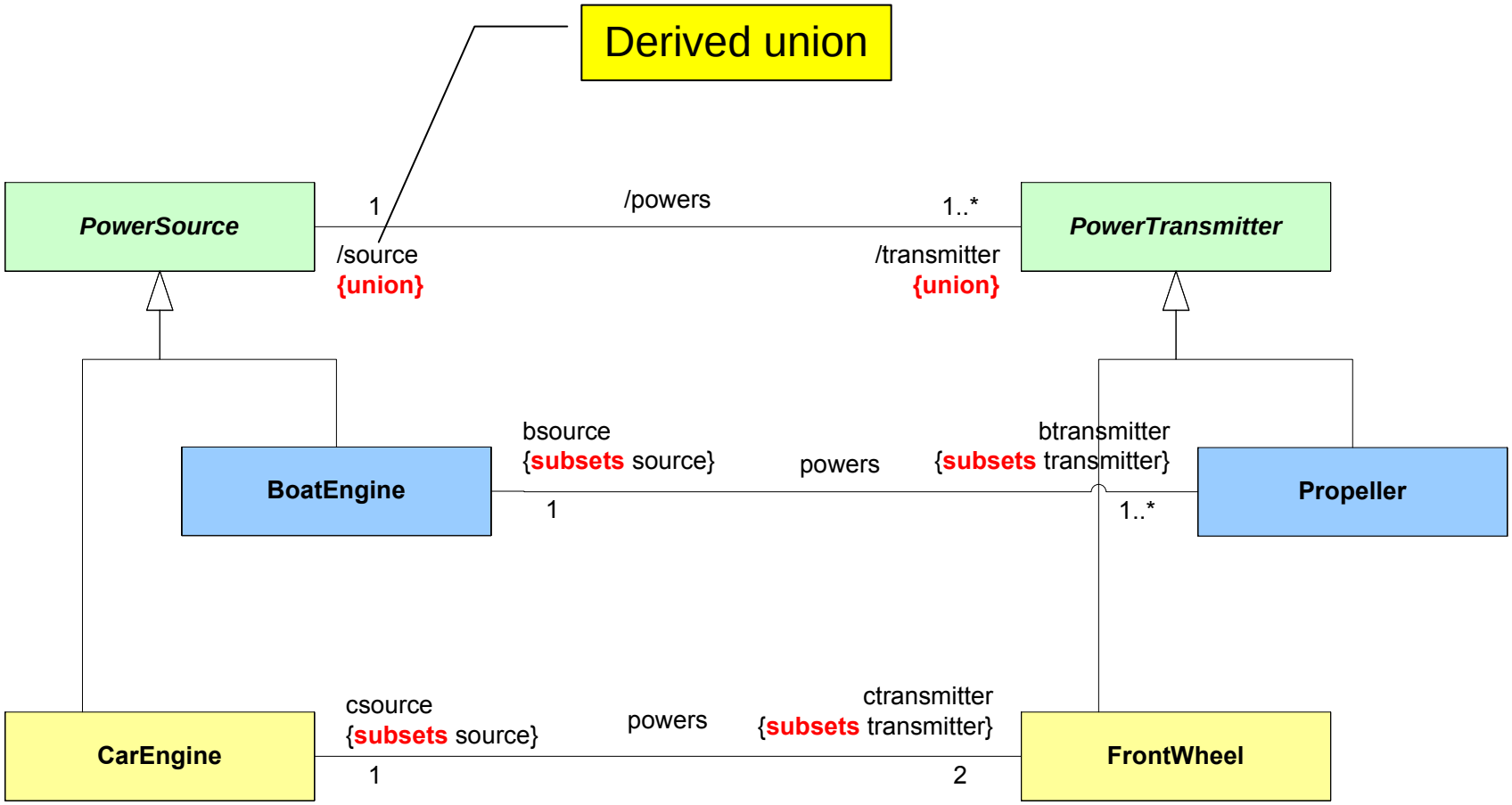


Incorrect

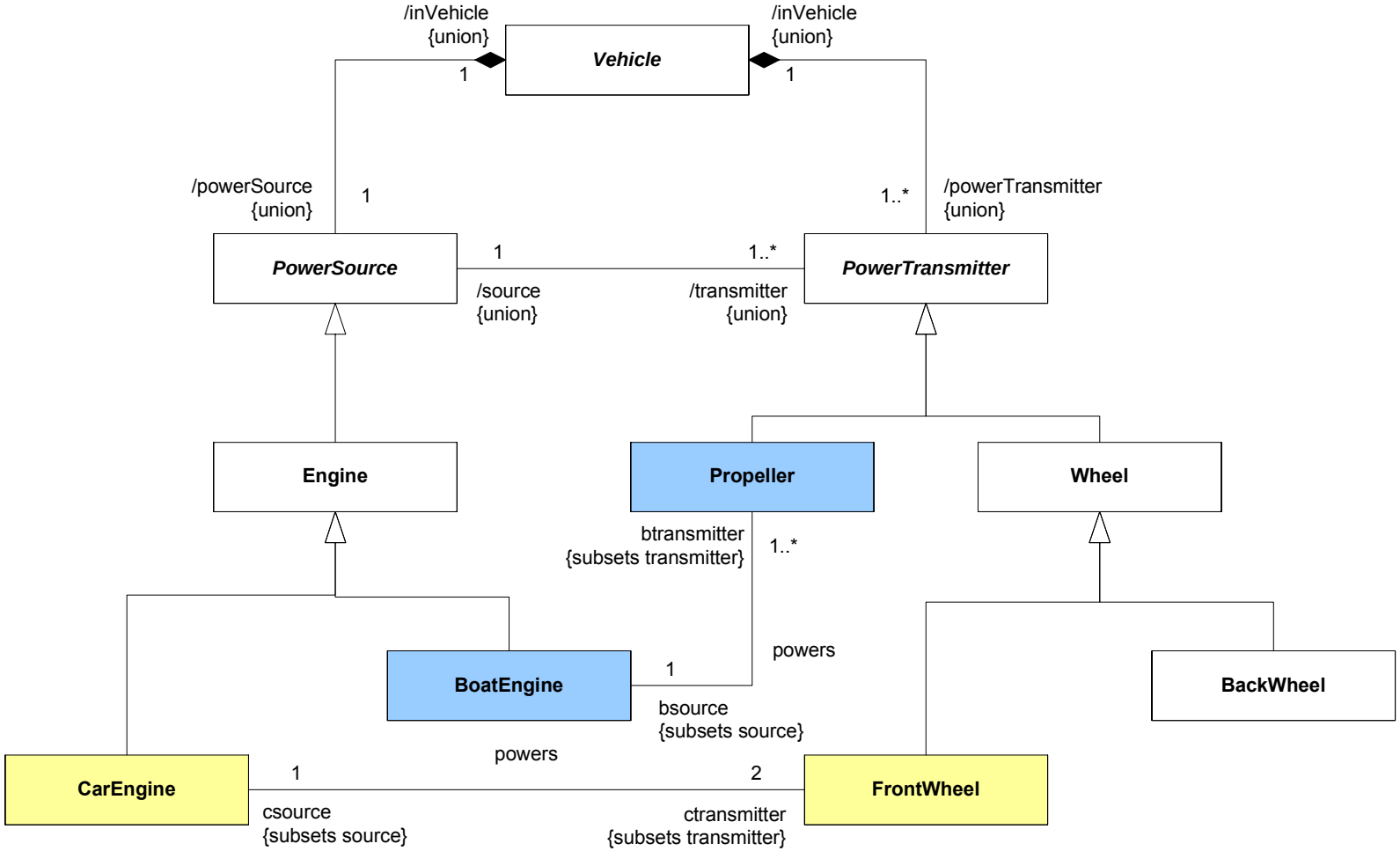
Redefines/Subsets revisited (3)



Redefines/Subsets revisited (4)

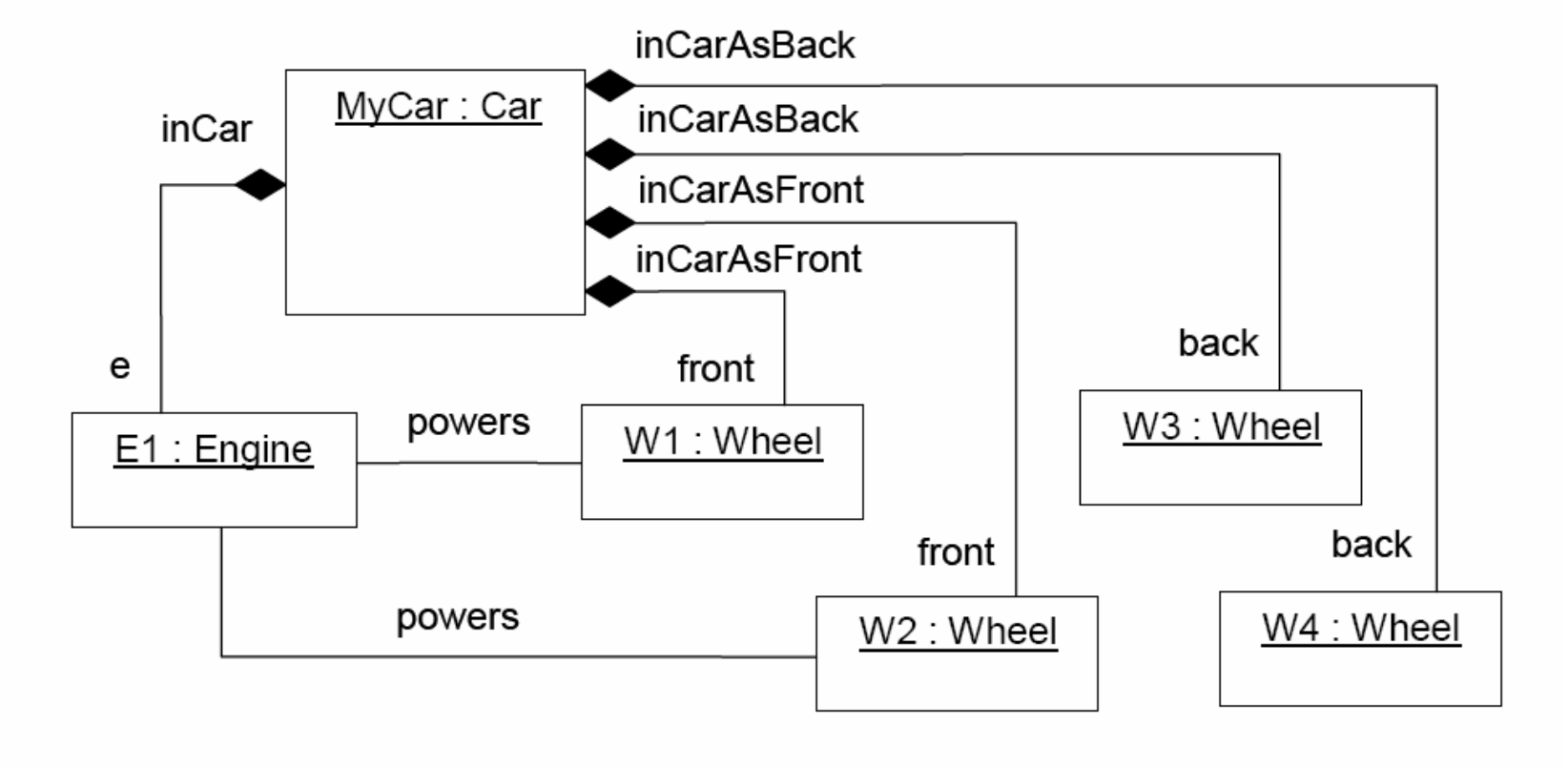


Redefines/Subsets revisited (5)



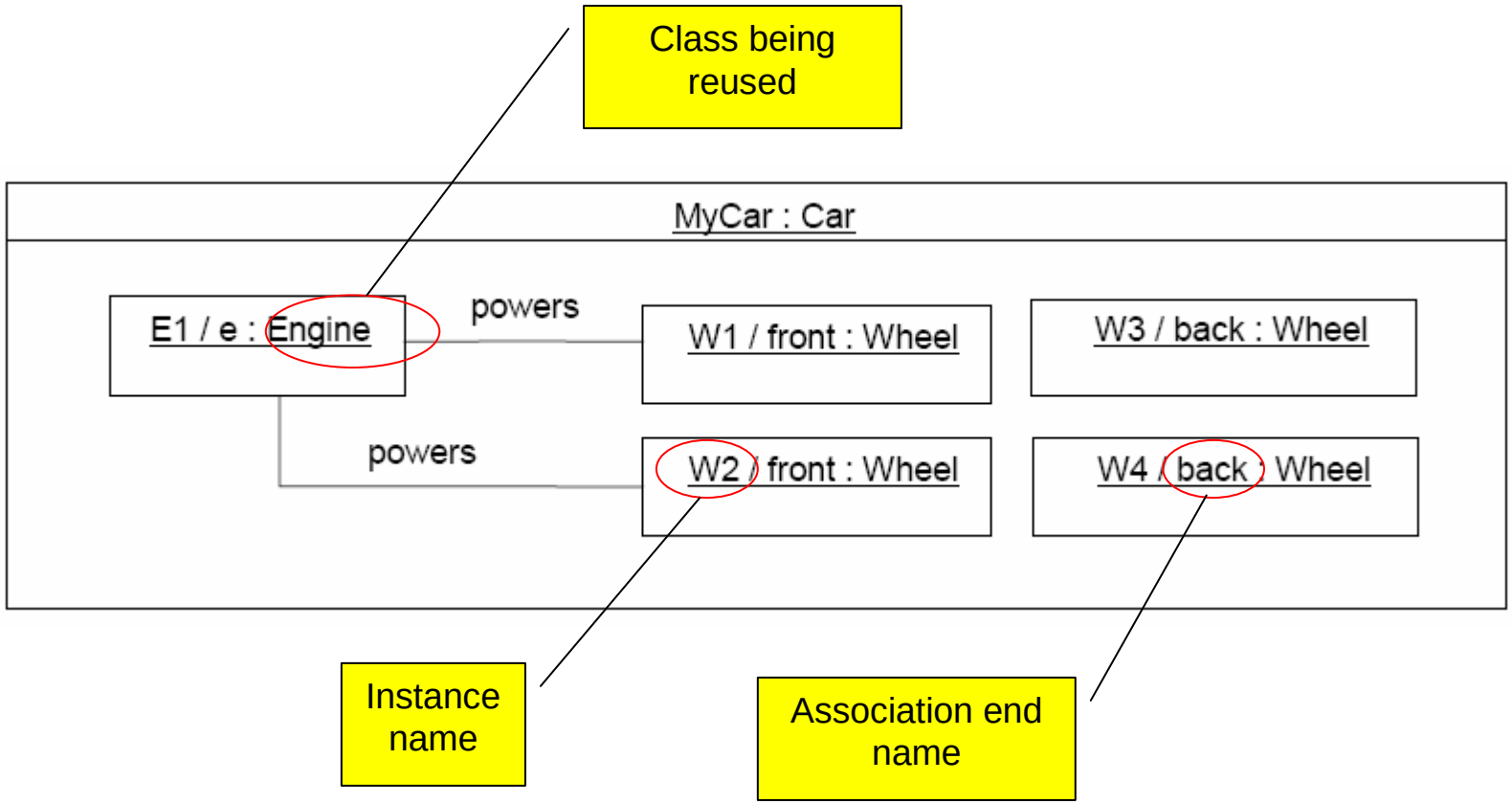
UML 2 Composition (3)

Instance model:



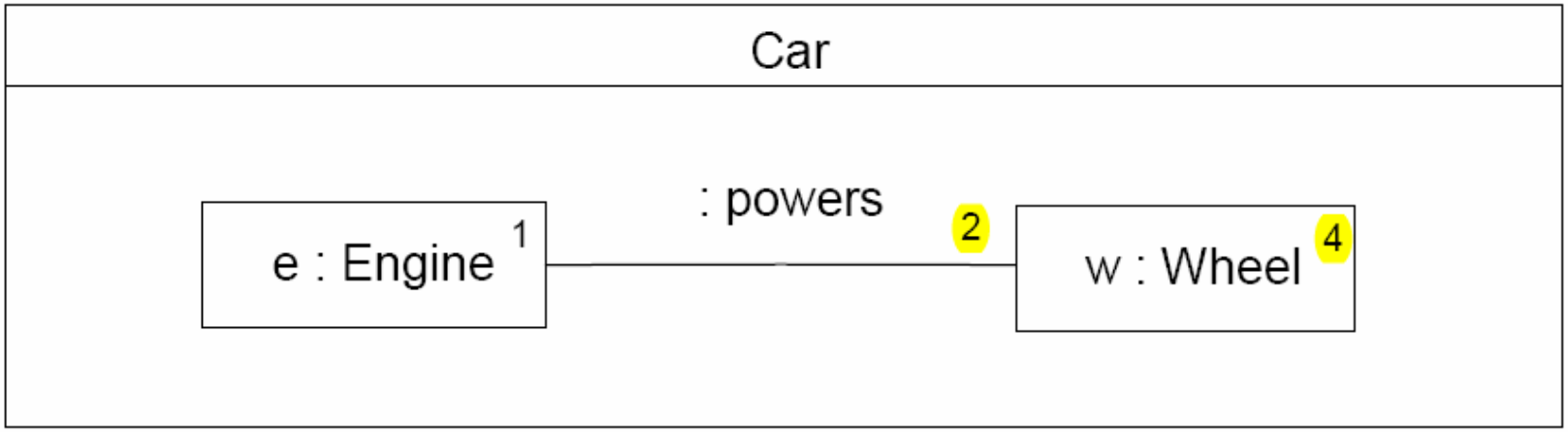
UML 2 Composition (4)

An alternate representation of an instance model:

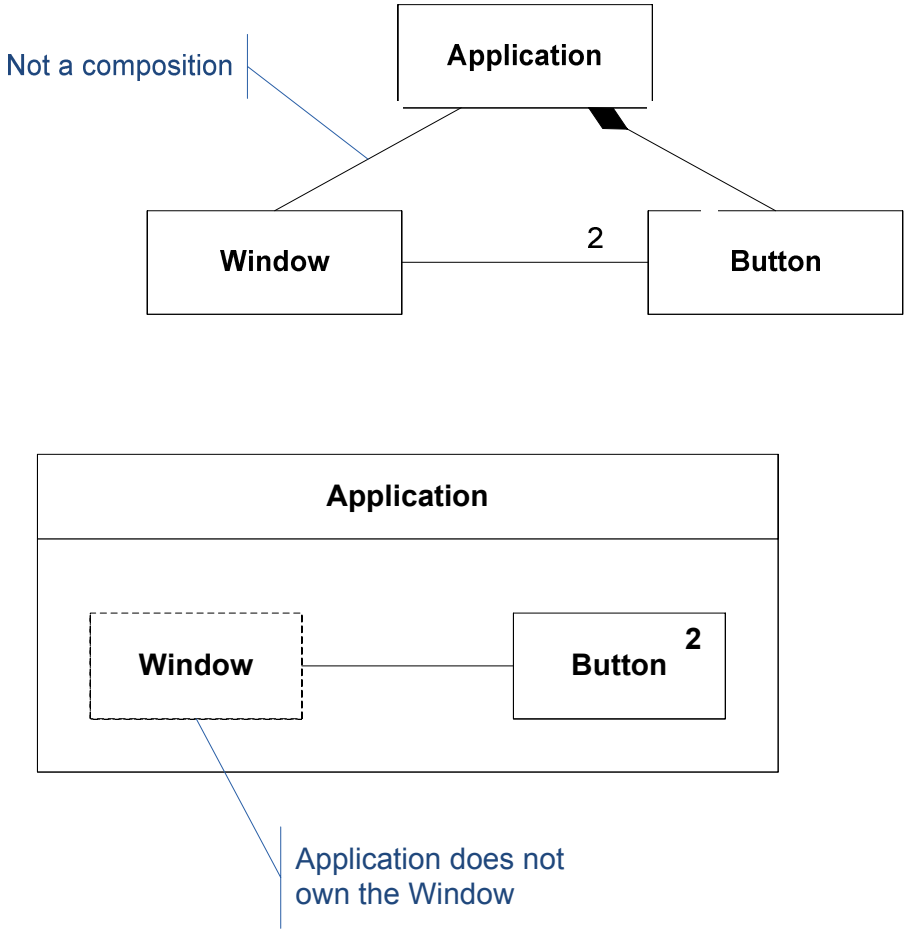


UML 2 Composition (5)

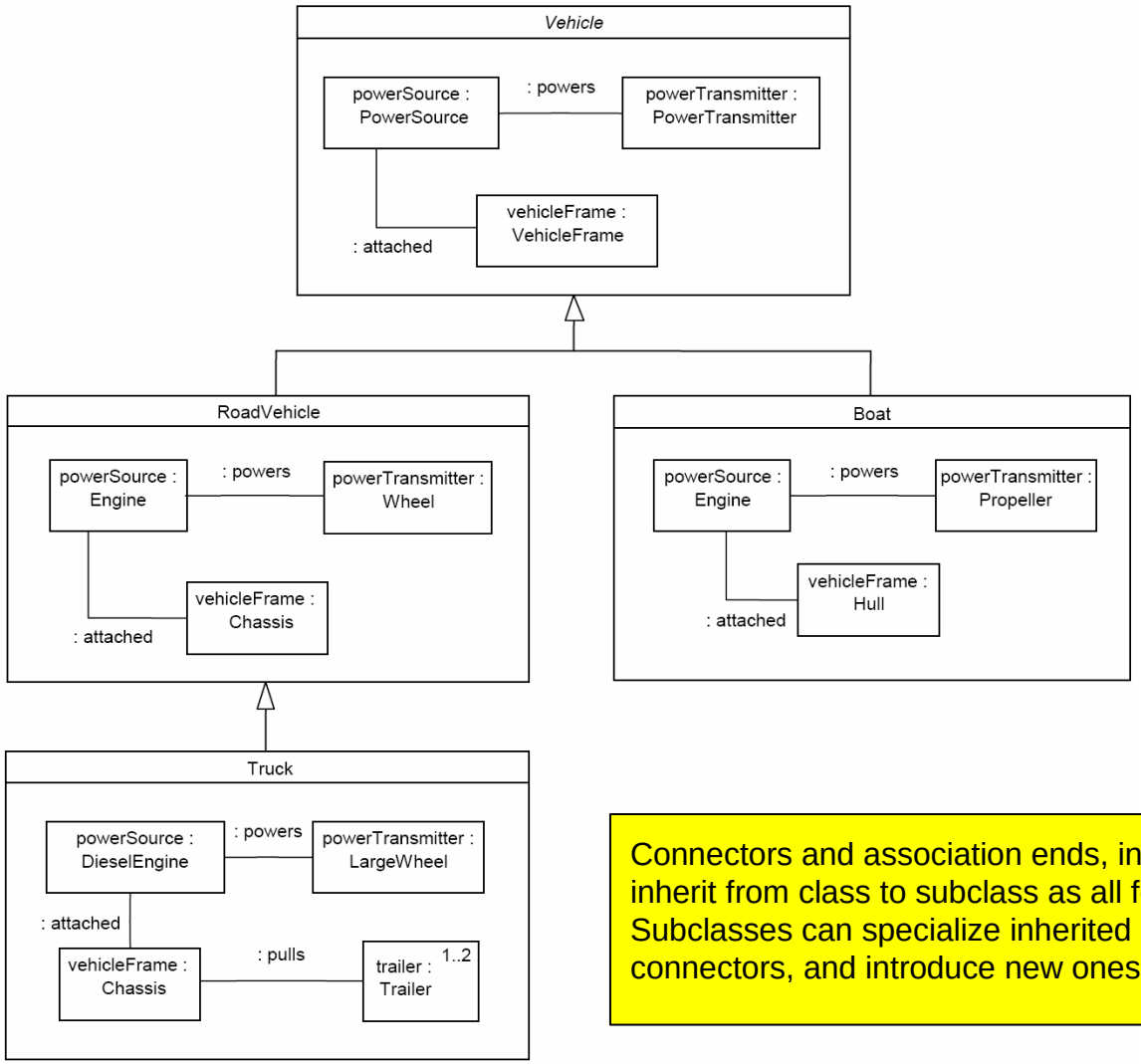
- Connectors can have multiplicities different from the association ends (parts) they relate. For example, a more general model of cars that covers both front-wheel drive and rear-wheel drive is shown in the Figure below. It uses a new association end *w* that navigates to all the wheels in a car. The new end has a multiplicity of 4, but the POWERS connector to it has a multiplicity of 2 on that end. This means for each car, two of the four wheels in the car will be



UML 2 Composition (6)



Inheritance of composite structure



Connectors and association ends, including parts, inherit from class to subclass as all features do. Subclasses can specialize inherited parts and connectors, and introduce new ones.

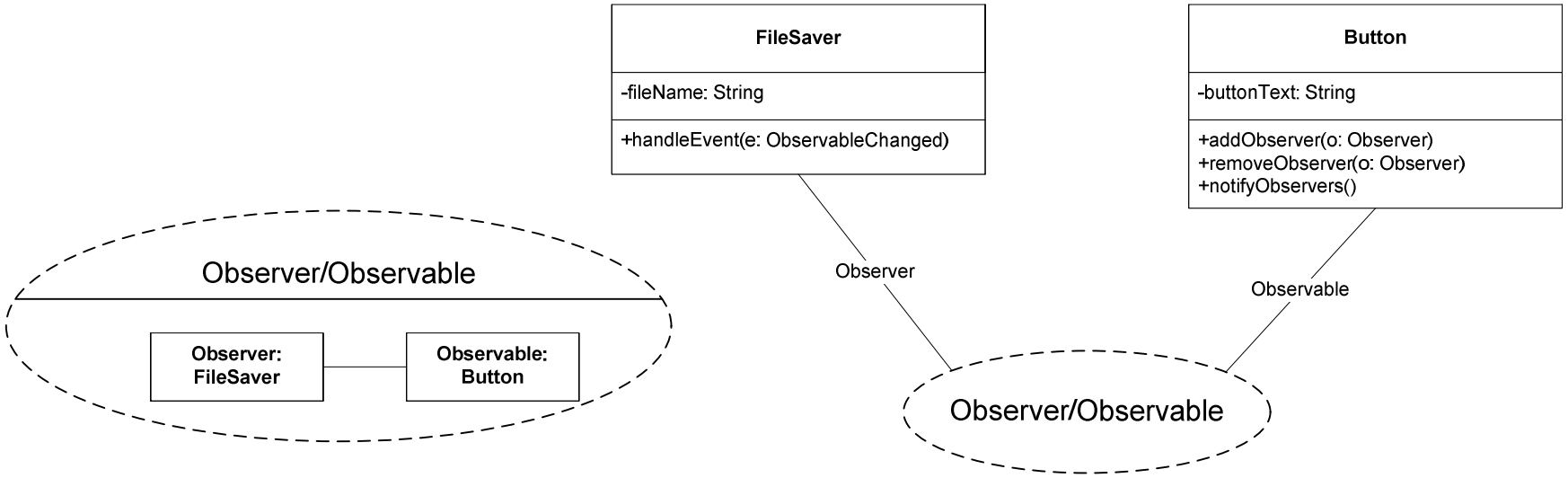
Collaborations (1)

- One of the primary purpose for composite structures is to document how a particular piece of functionality is implemented within a system.
- This organization of elements to realize behavior is called a Collaboration.
- A Collaboration = collection of instances wired together using connectors to show the communication flow.
- Within a Collaboration, it is helpful to name the instances involved.

  Typically named based on its role in the collaboration.
- Representation: a dashed ellipsis with the name of the collaboration written inside.

Collaborations (2)

- Details of a collaboration:

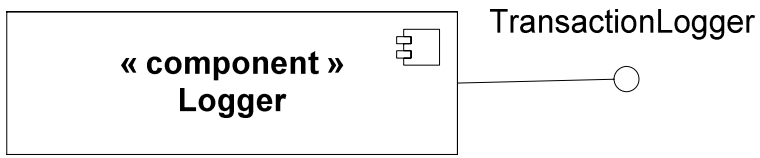


- Collaboration Uses

  A collaboration use represents one particular use of a collaboration to explain the relationships between the properties of a classifier. A collaboration use shows how the pattern described by a collaboration is applied in a given context, by binding specific entities from that context to the roles of the collaboration.

Component

- [UML-Glossary] Component = A modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. As such, a component serves as a type, whose conformance is defined by these provided and required interfaces (encompassing both their static as well as dynamic semantics).
- A Component is a kind of Classifier; it can have methods that realize interfaces, have statecharts and use cases.
- Components are coarse-grained elements that are usually replaced as a whole in the application.
- Component-based Development approaches use the metaphor of construction through assembly rather than construction via invention.



Subsystems

- [UML-Glossary] Subsystem = A unit of hierarchical decomposition for large systems. A subsystem is commonly instantiated indirectly. Definitions of subsystems vary widely among domains and methods, and it is expected that domain and method profiles will specialize this construct. A subsystem may be defined to have specification and realization elements.
- A subsystem is a stereotype of both Package and Classifier.
- A subsystem is instantiable.
- A subsystem is used to organize the run-time system consisting of instances.
- The real work of a subsystem is implemented by the run-time instances contains within the subsystem. It offers up the collaborative behavior of the contained elements.
- Subsystems have three aspects: operation, specification, and implementation. The operations are the set of services directly offered by the Subsystem.