Object-orientation as Composite Structure:

(Onto)Logical Object-orientation

Conrad Bock,
U.S. National Institute of Standards and Technology

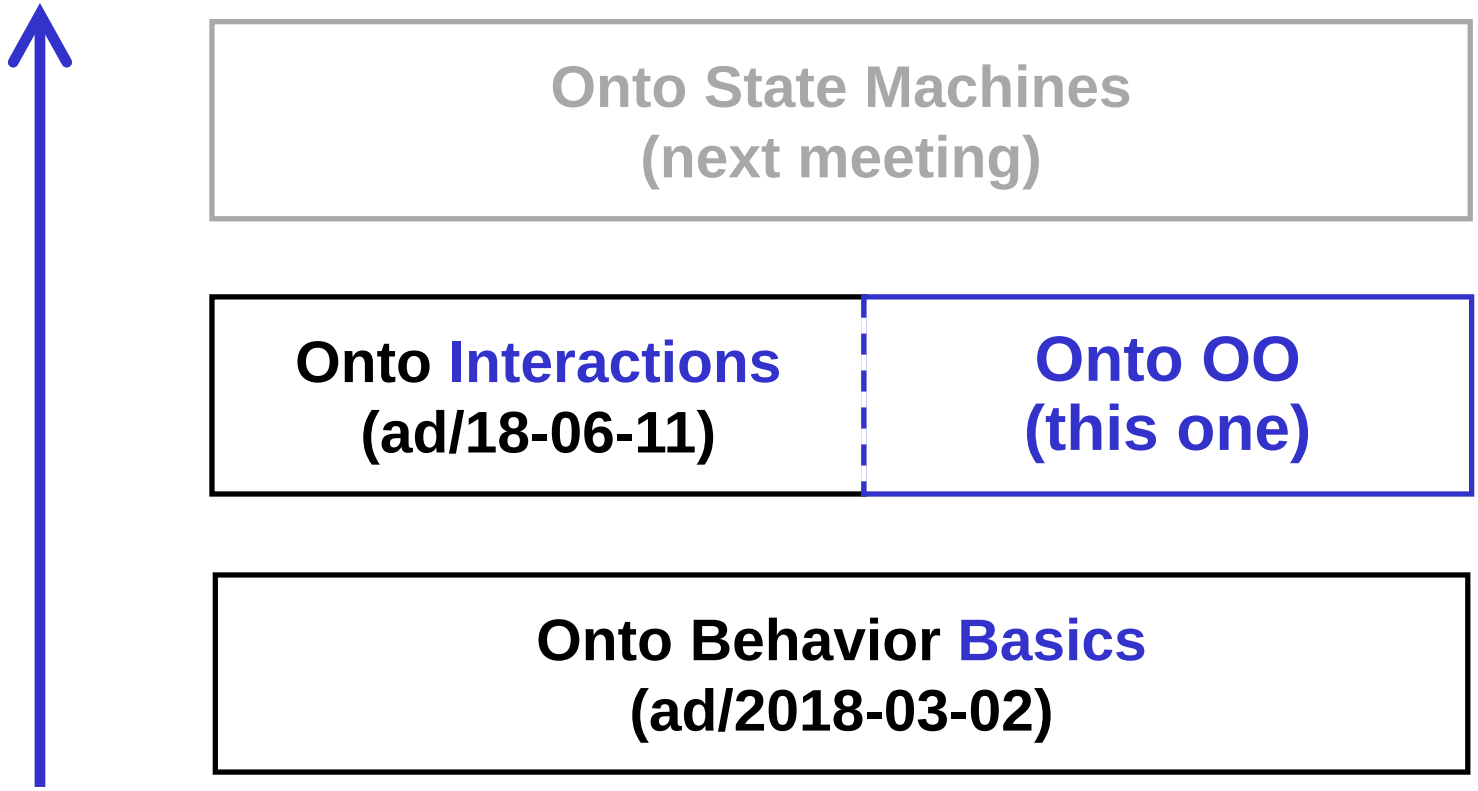
Charles Galey
Lockheed Martin

NIST
National Institute of Standards and Technology

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary

Behavior as Composite Structure Presentation Stack



Overview

- RoadMap
- **Motivation**
  - **Behavior, review**
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary

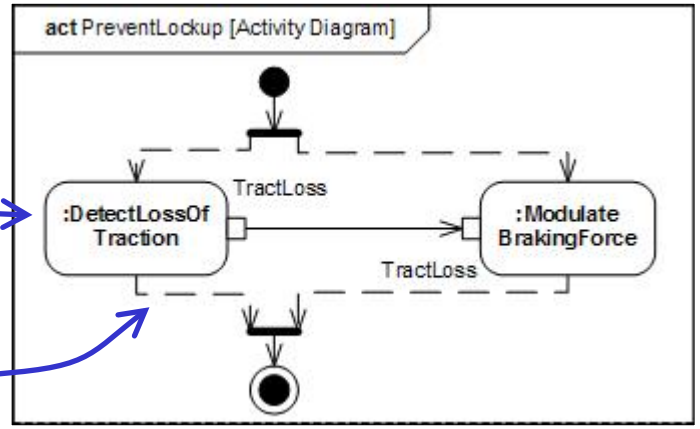
General Problem

- **UML has three behavior diagrams.**
  - **Activity, state, interaction.**
- **Very little integration or reuse between them.**
  - **Three underlying metamodels.**
  - **Three representations of temporal order.**
- **Triples the effort of learning UML and building analysis tools for it.**

General Solution

- **Treat behaviors as assemblies of other behaviors.**
  - **Like objects are assemblies of other objects.**
- **Assembly = UML internal structure**
  - **Pieces represented by properties.**
  - **Put together by connectors.**
- **Put all behavior diagrams on the same underlying behavior assembly model.**

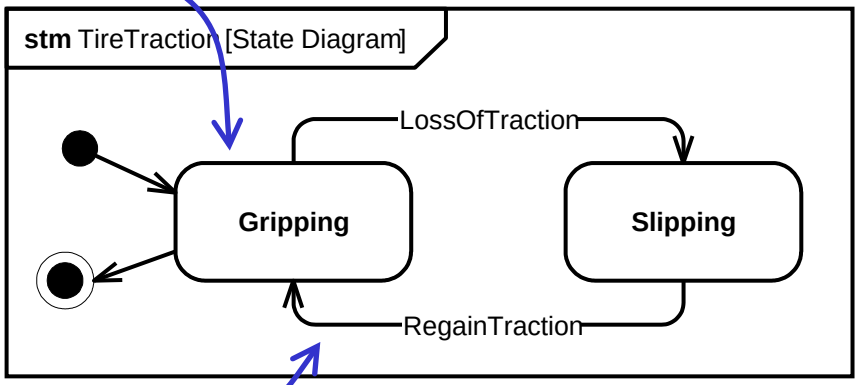
Behaviors as Composite Structure



Property

Connector

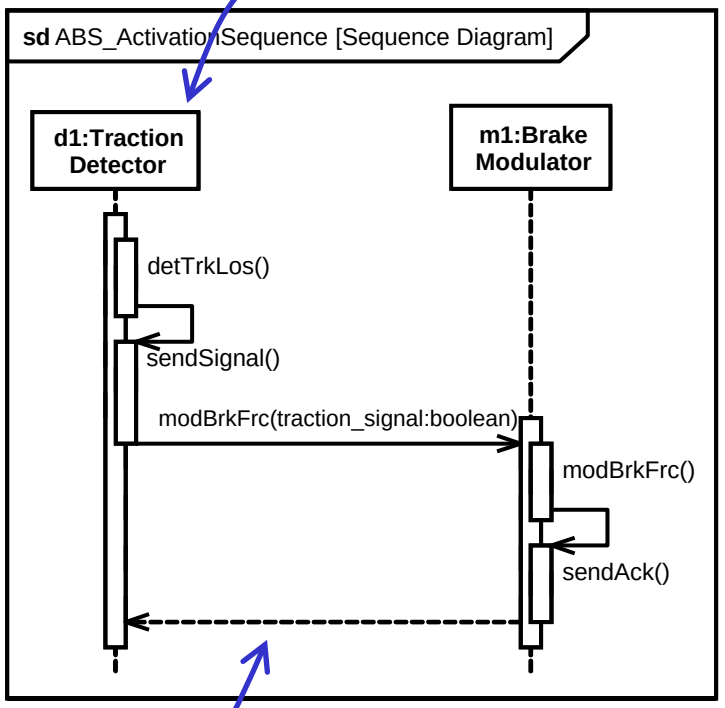
Activity



Property

Connector

State Machine



Property

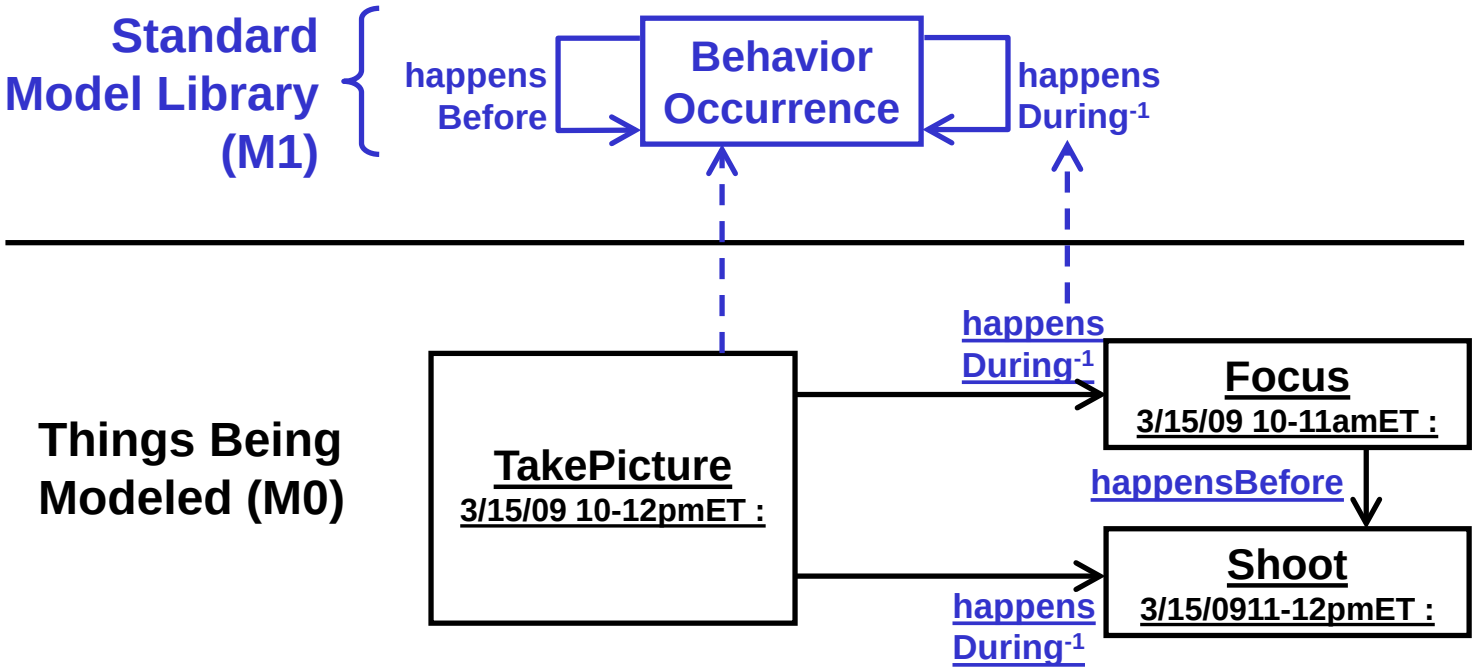
Connector

Interaction

Behavior: What's Being Modeled?

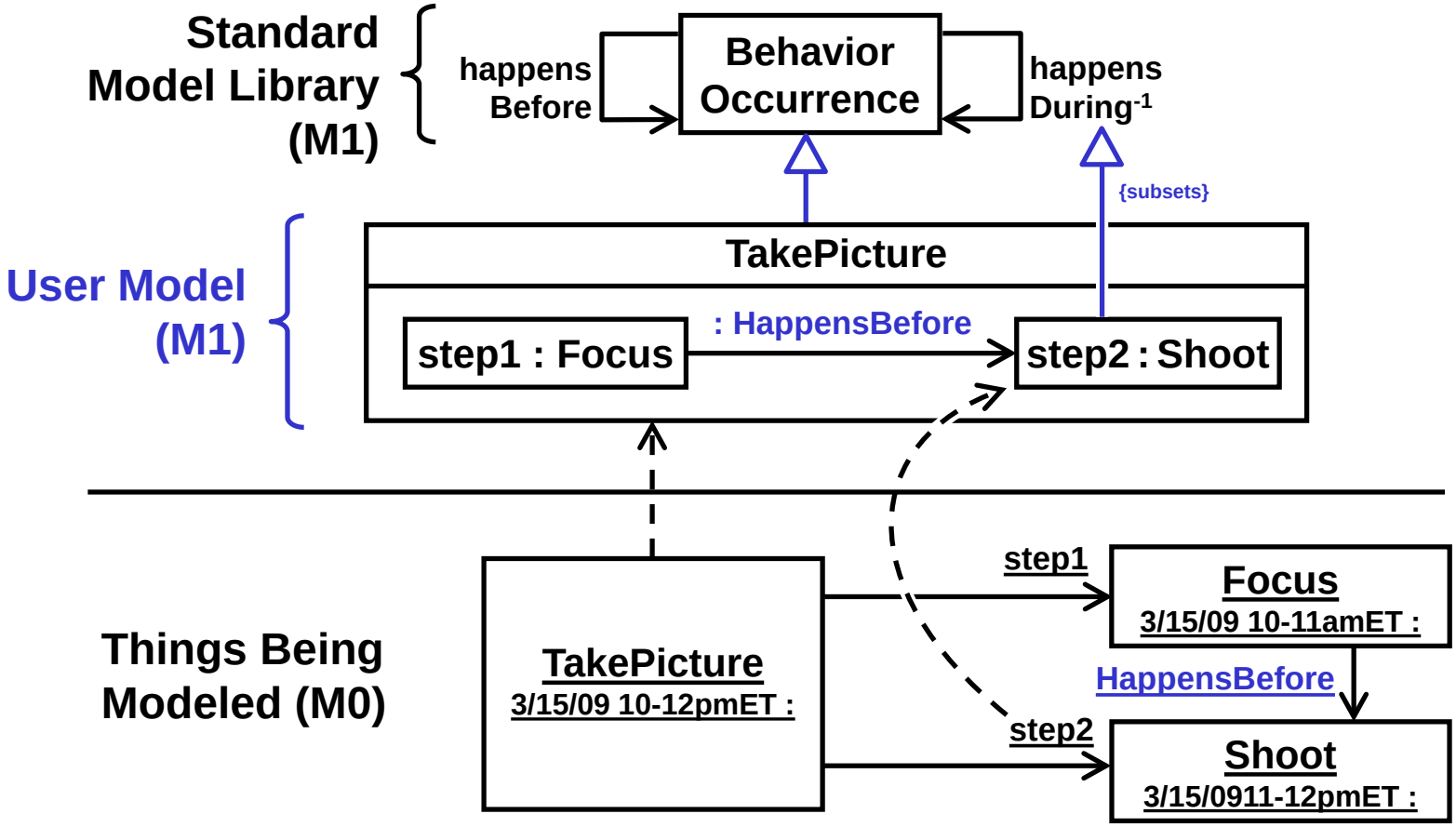
- "Things" that occur in time
  - Eg, taking a picture, focusing, etc.
  - Not "behaviors", "actions", etc.

Behavior: What's in Common?



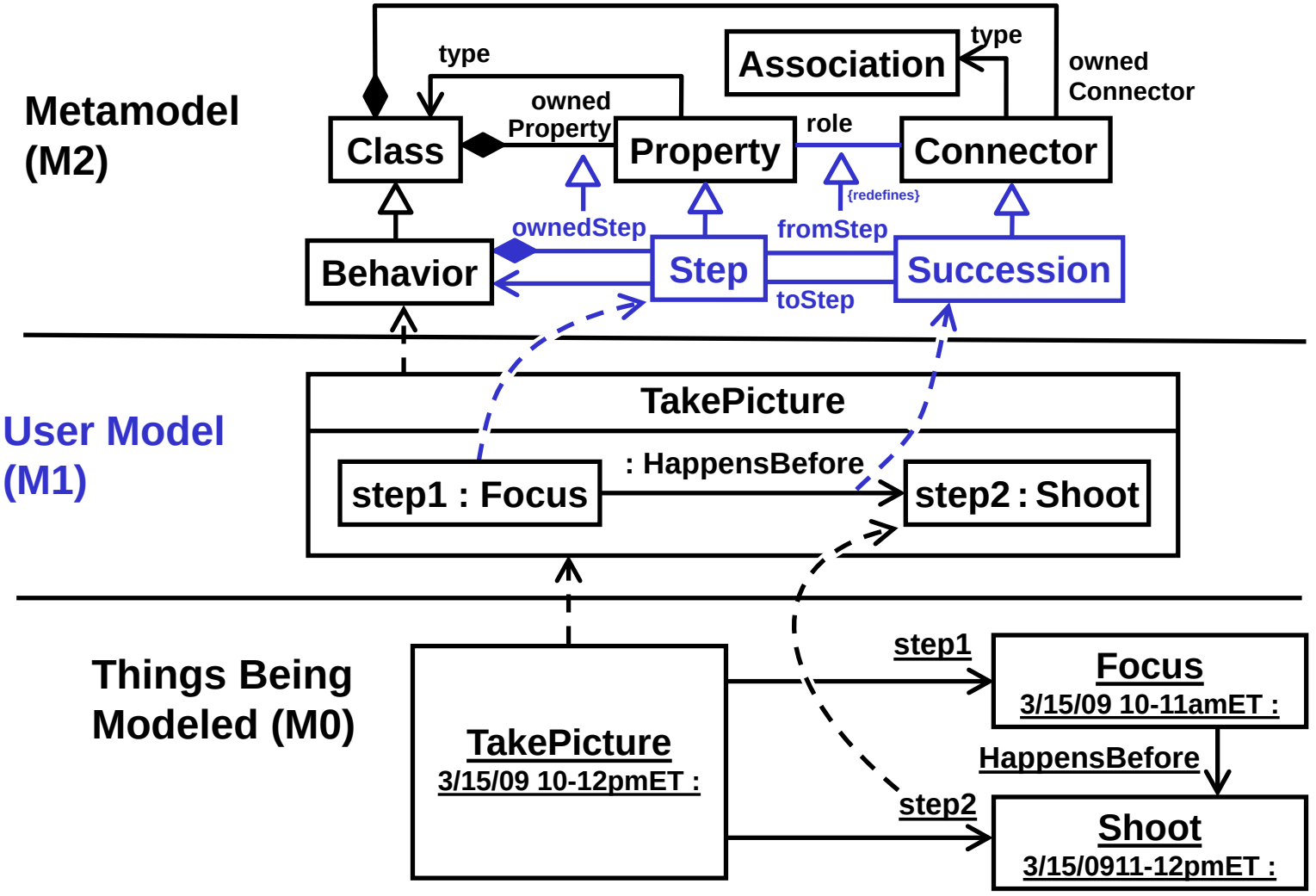
- They happen before or during each other.
  - Construct M1 library for this.
  - Use it to classify things being modeled.

Behavior: Use Library



- **Specialize library classes and subset/redefine library properties.**

Behavior: Too repetitive at M1?

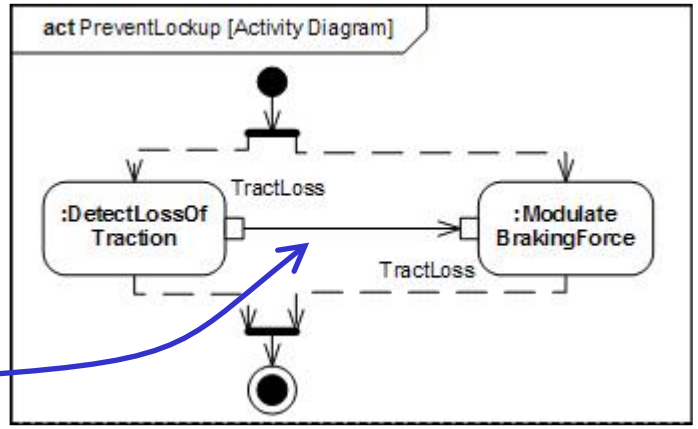


- **Capture M1 patterns in M2 elements.**
  - **Tools apply patterns automatically.**

Overview

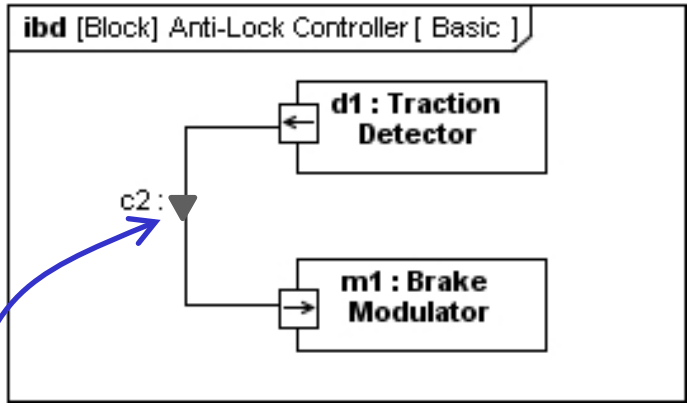
- RoadMap
- Motivation
  - Behavior, review
  - **Interactions, review**
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary

Interactions Problem



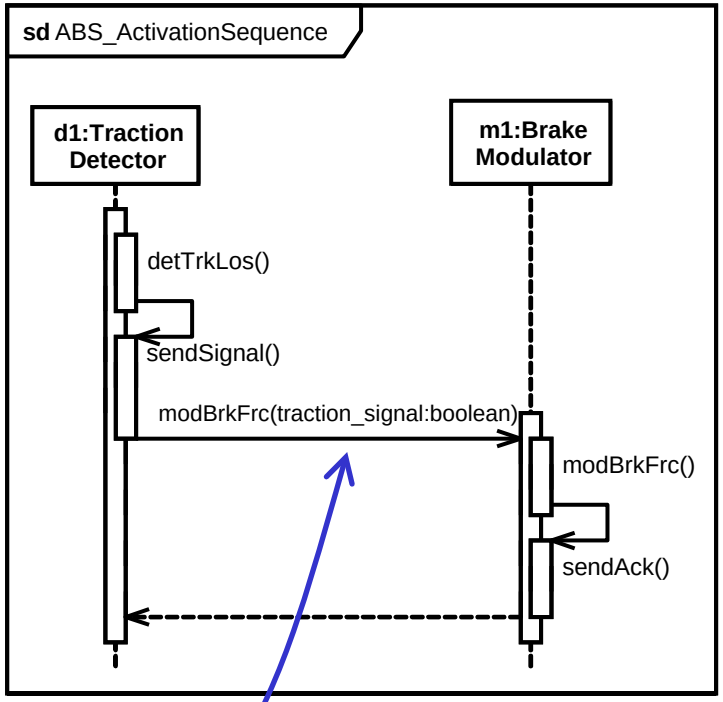
Object Flow

Activity



Item Flow

SysML Internal Block Diagram



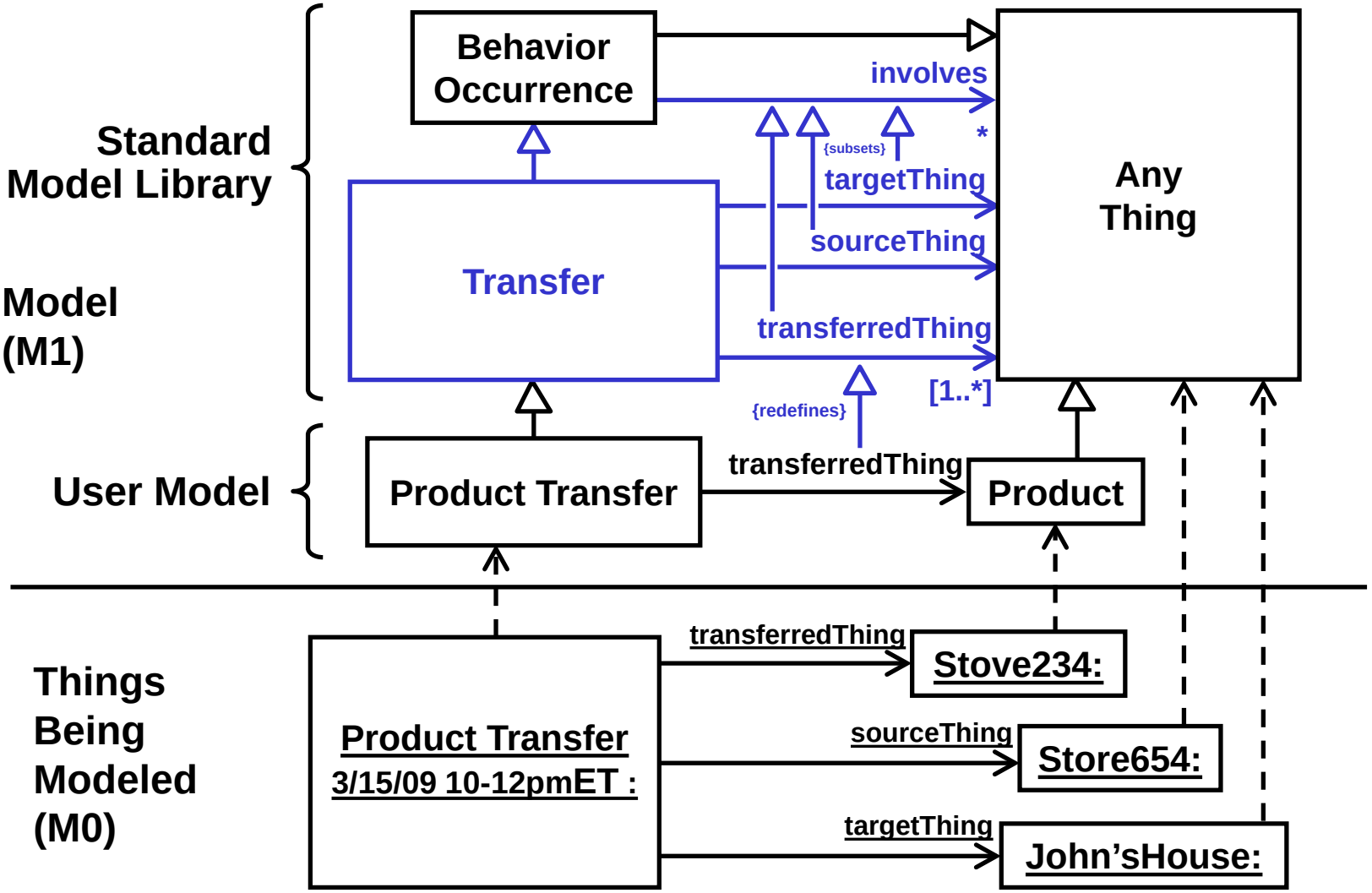
Interaction

Message

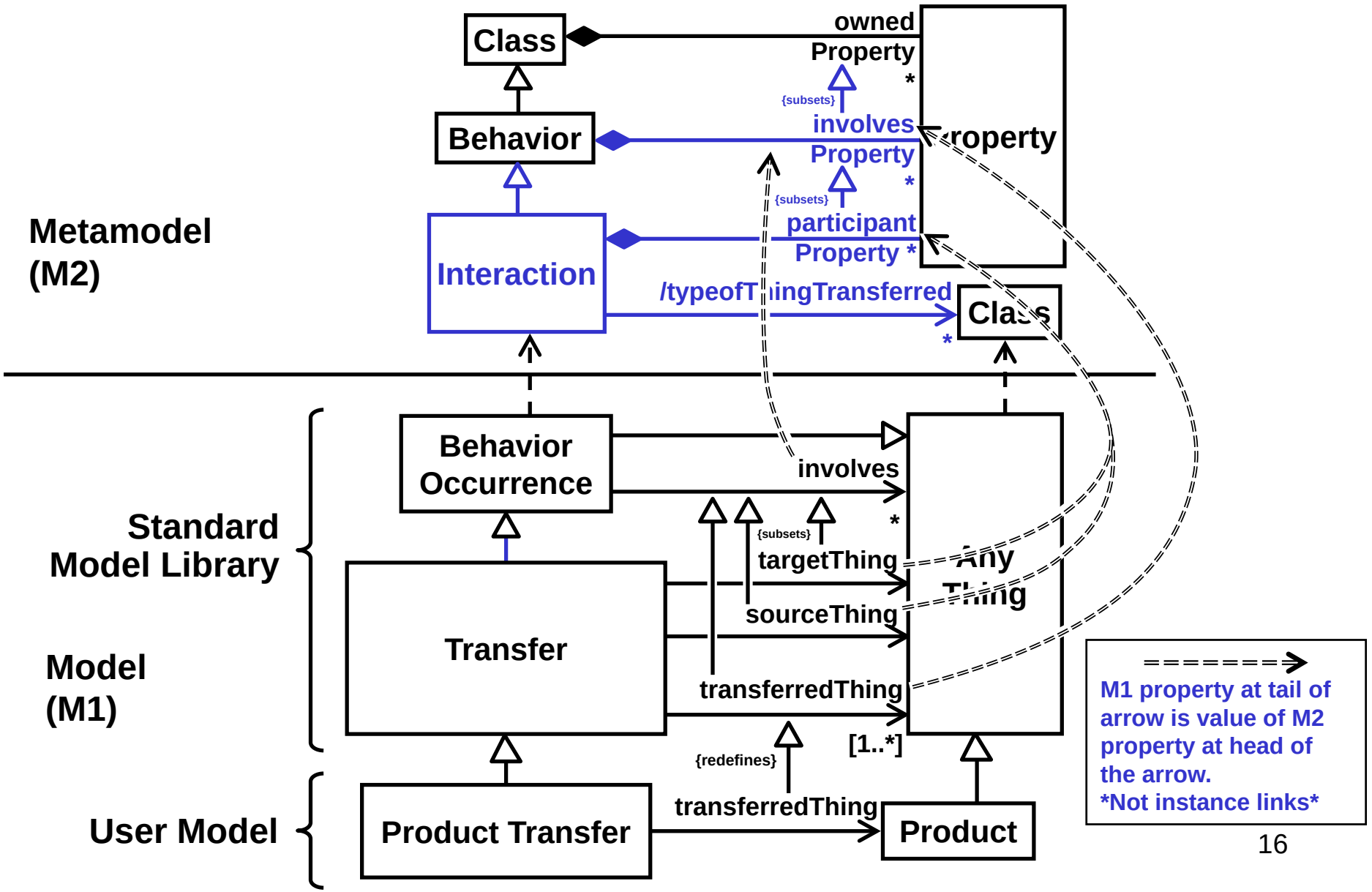
Interactions Requirements

1. **Between things that outlive interactions.**
   - **Objects have many interactions over time.**
   - **Not just between steps in an activity.**
2. **Interactions are reusable and composable.**
   - **The same kind of interaction might be used in many user models and**
   - **contain many other interactions ordered in time.**
3. **Interacting objects have "mailboxes".**
   - **Things being exchanged leave and arrive at specified places in the interacting objects.**
   - **Aka, output/inputs.**

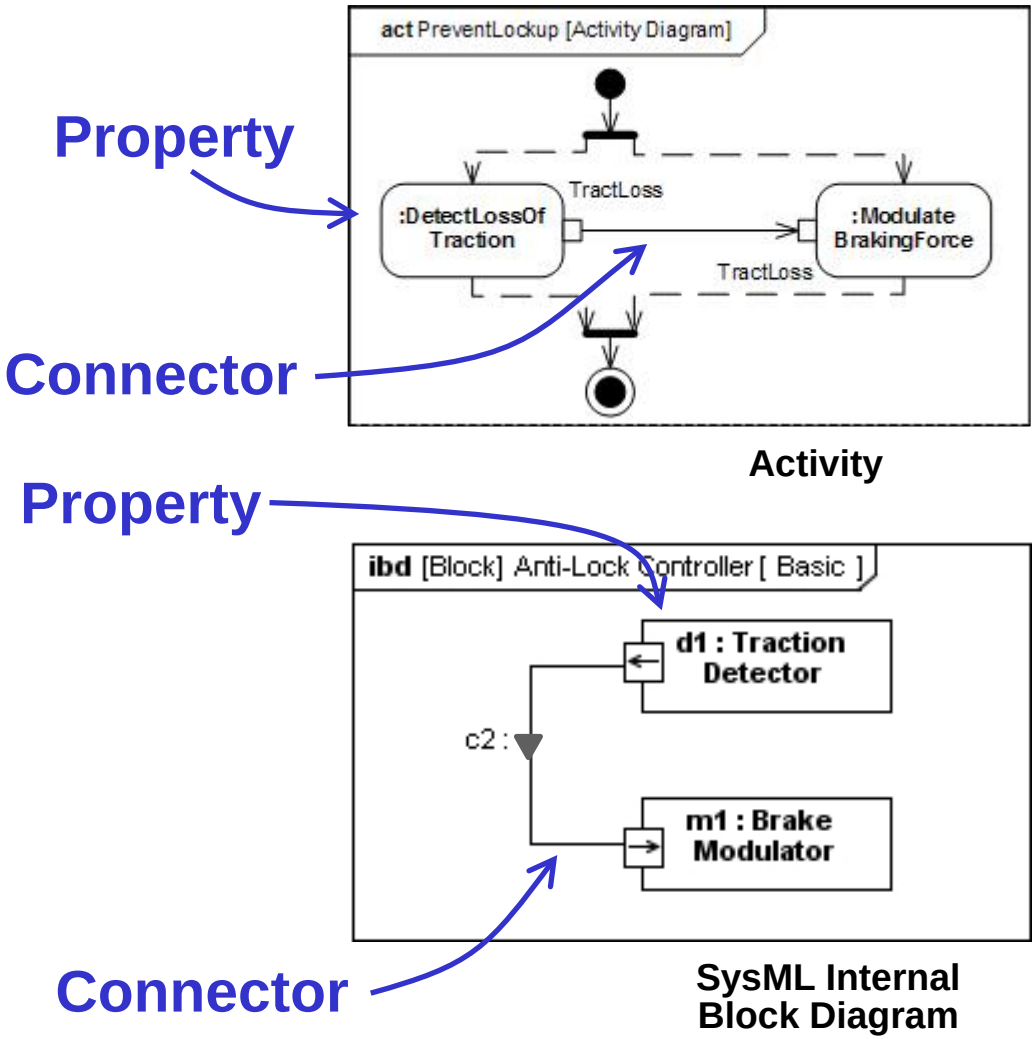
Transfers (M1)

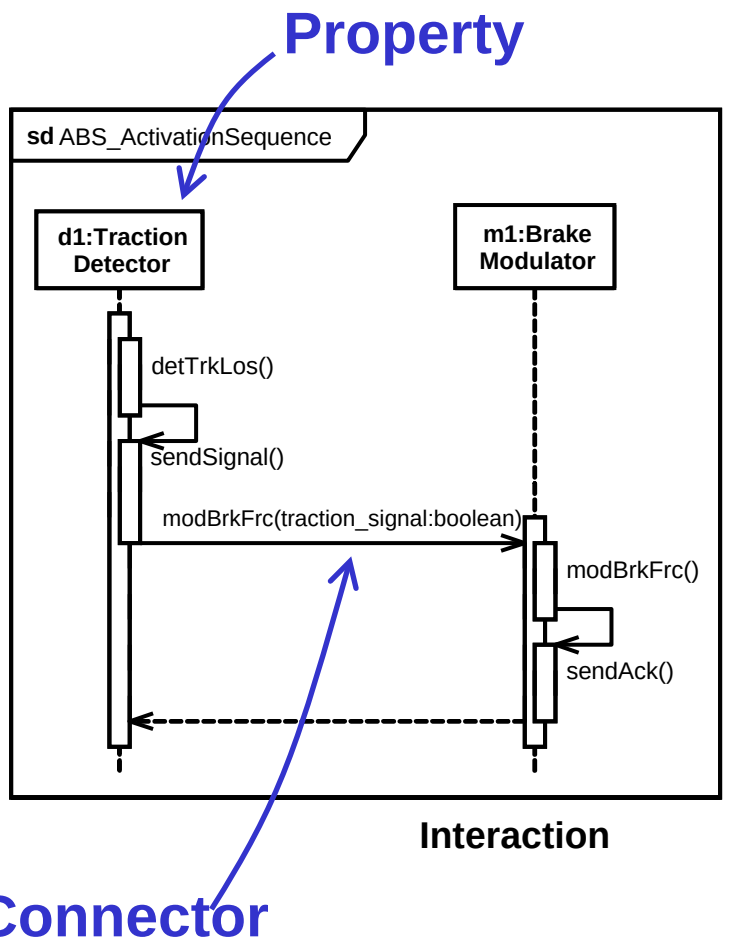


Interactions (M2)



Transfers Along Connectors?



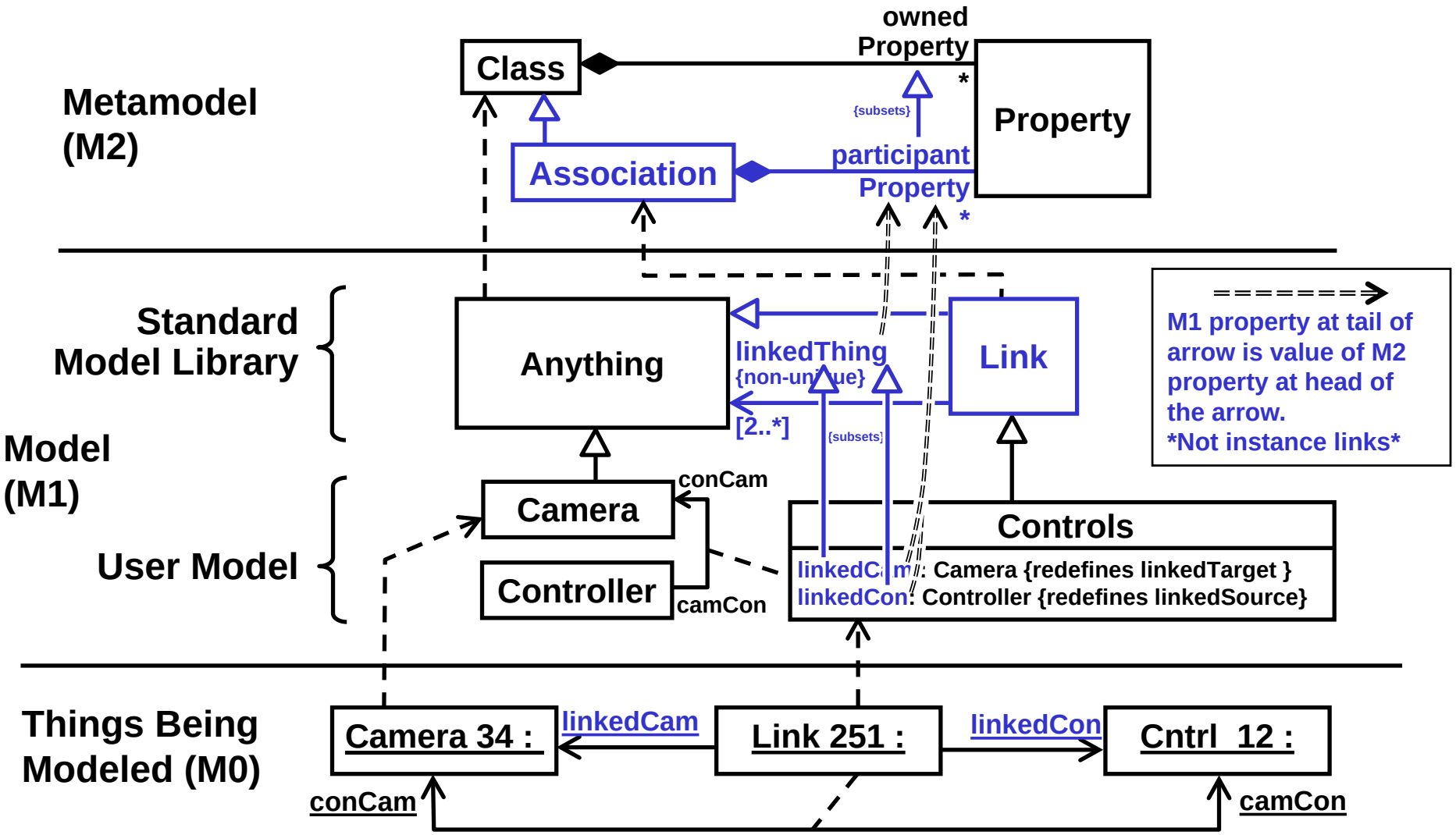


- **Connectors are typed by associations.**
  - **But transfers are behaviors.**

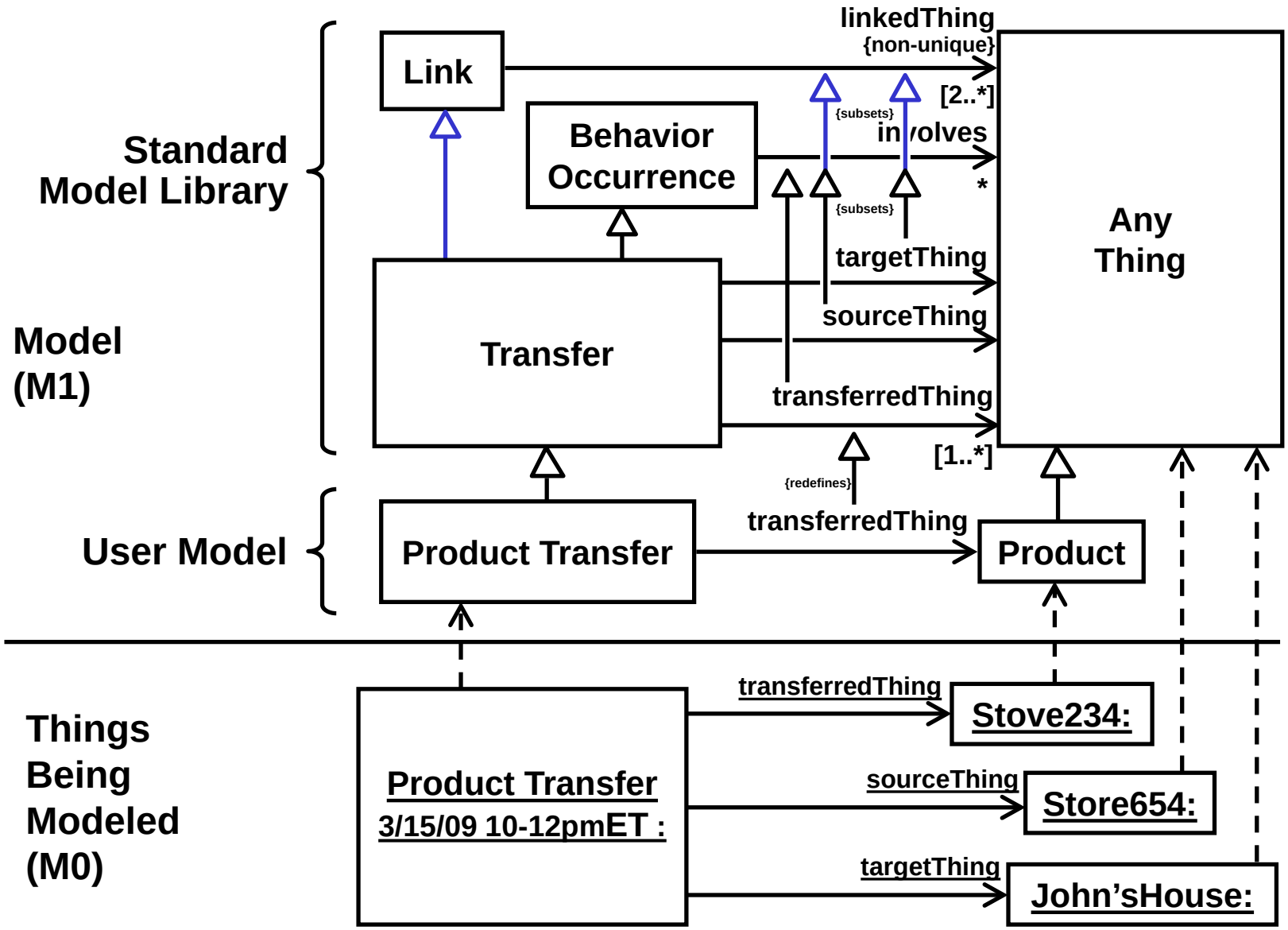
Interaction = Behavior & Association

- **Associations and behaviors both have objects that participate in them.**
  - **Associations link their participants.**
  - **Behaviors involve their objects.**
    - **Interactions have lifelines.**
    - **Activities have object nodes, partitions, etc.**
    - **Behaviors have parameters.**
- **Interactions are behaviors that are also associations between their participants.**

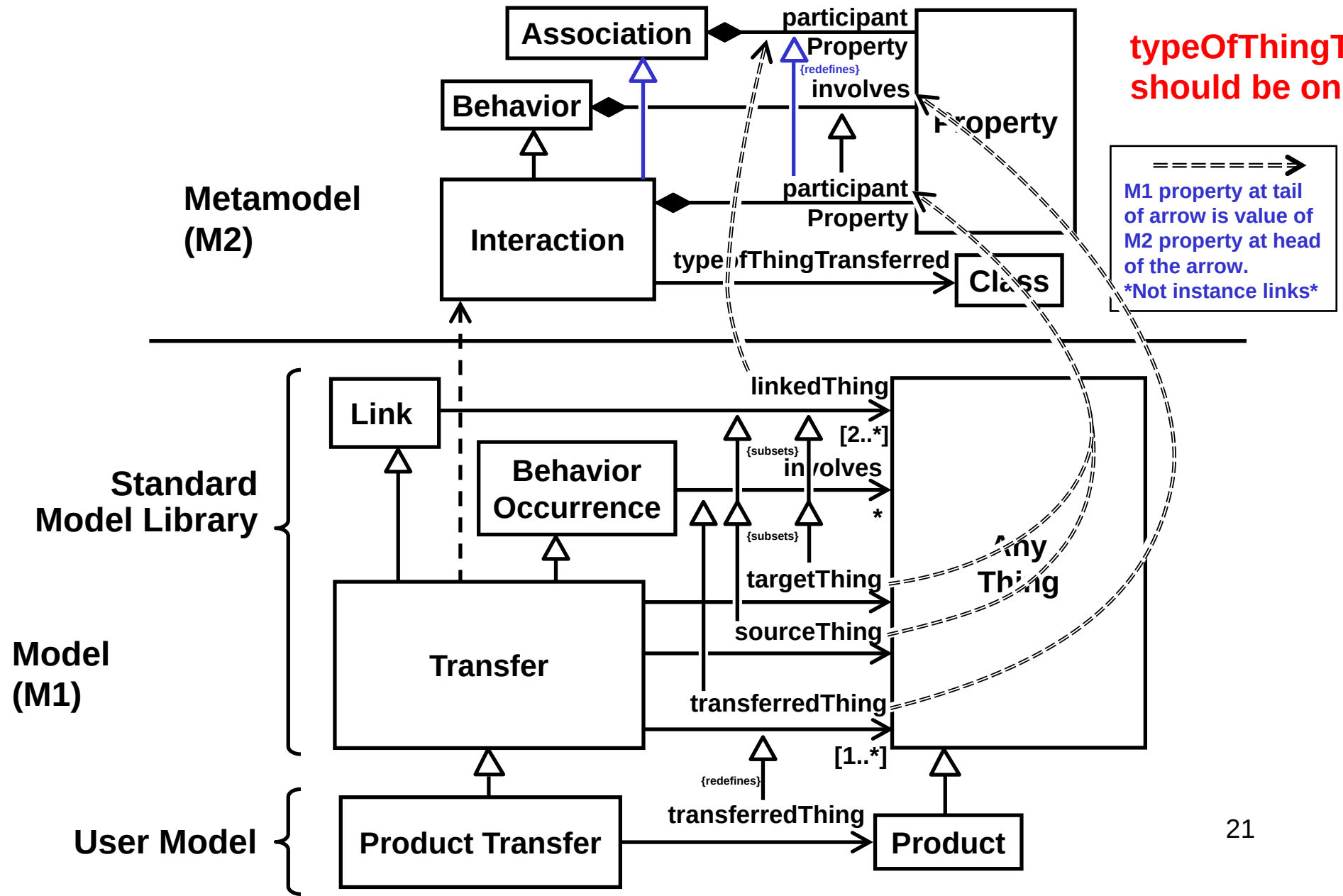
Links (M1) & Associations (M2)



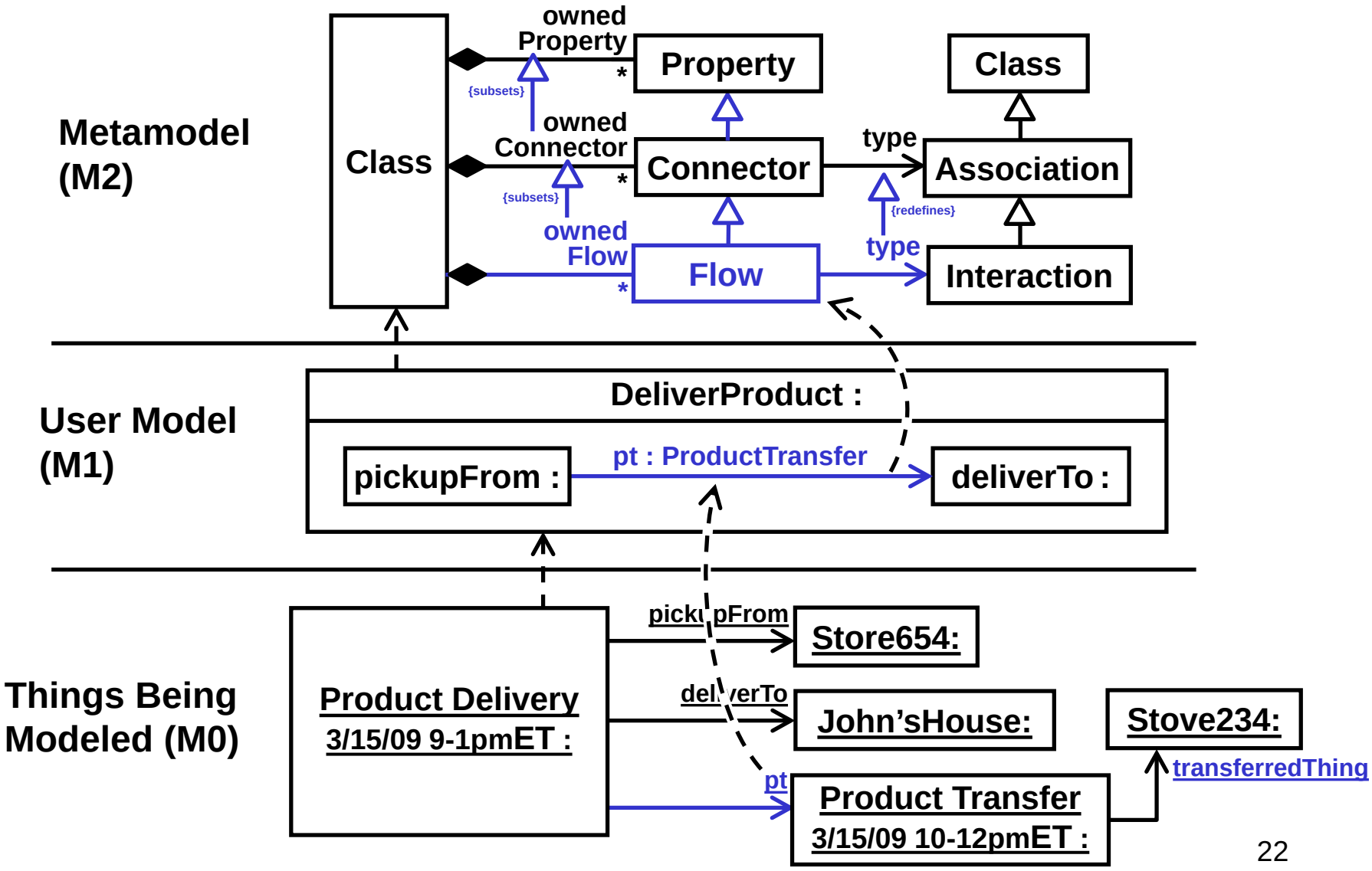
Transfers as Links (M1)



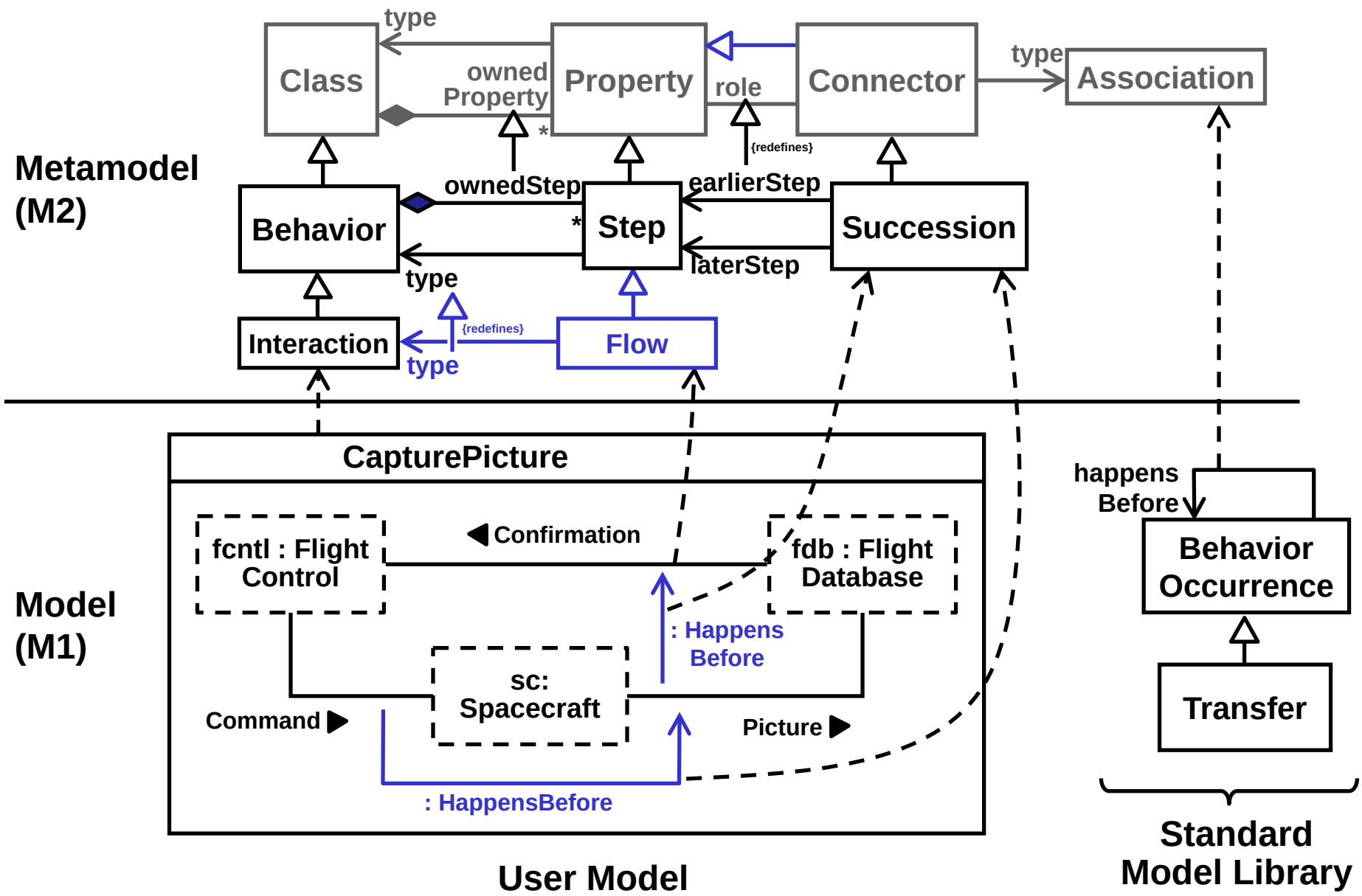
Interaction Participants (M2)



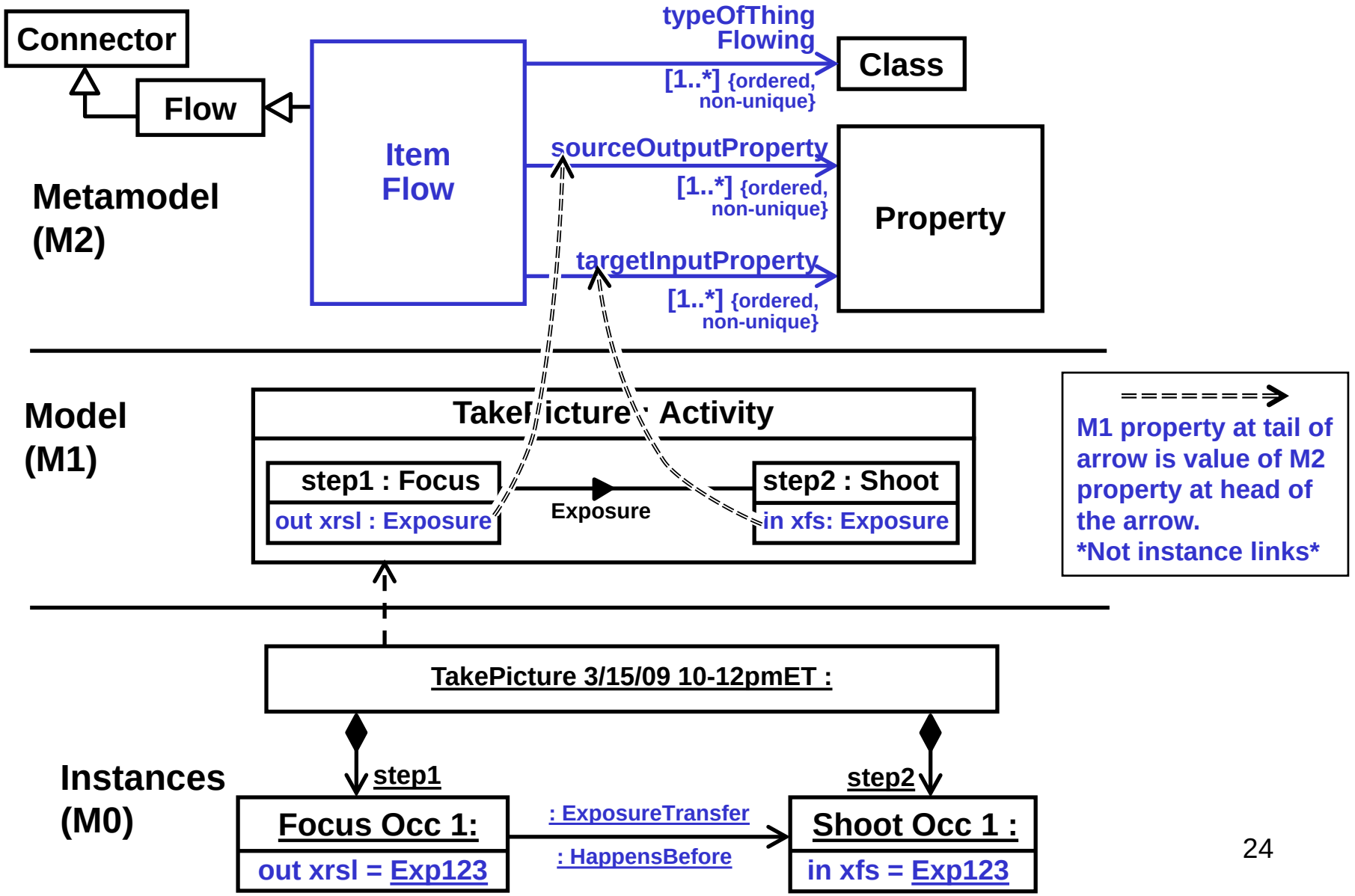
Connectors Reusing Interactions



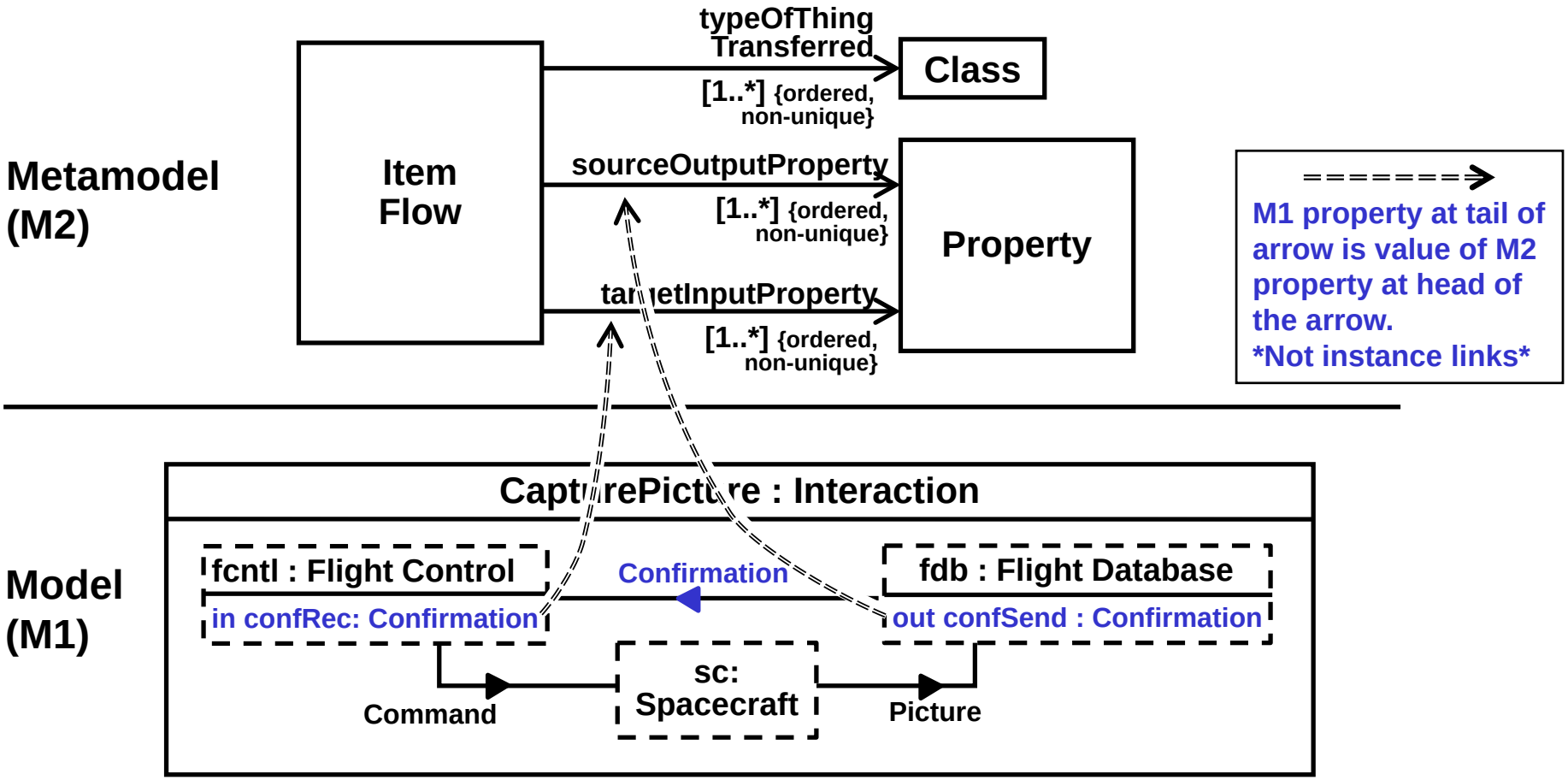
Flow Steps


Metamodel
(M2)
Class
type
owned
Property
*
Property
role
Connector
type
Association
{redefines}
Behavior
ownedStep
*
Step
earlierStep
laterStep
Succession
type
Interaction
{redefines}
type
Flow
Model
(M1)
CapturePicture
fcntl : Flight
Control
Confirmation
fdb : Flight
Database
: Happens
Before
sc:
Spacecraft
Command
Picture
: HappensBefore
User Model
happens
Before
Behavior
Occurrence
Transfer
Standard
Model Library

Flows & Out/Inputs (OF)



Flows & Out/Inputs (FP)



Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - **OO behavior, requirements**
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary

OO Problems in UML/SysML

- **Encapsulated and “surfaced” behaviors modeled differently.**
  - **Namespace ownership for encapsulated behaviors (methods).**
  - **Operations for surfaced behaviors.**
- **Method specialization (“override”) doesn’t use generalization / inheritance.**

OO Problems in UML/SysML

- **Interfaces (service "bundles")**
  - **Missing supported interactions.**
    - **Expected order of operation calls, signal receipts, flowing property values.**
  - **Redundantly specified on both ends of interactions (eg, conjugation).**
  - **Need ports to distinguish interfaces uses.**
  - **Redundant model of behavior abstraction**
    - **Specify input/outputs of surfaced behaviors (ie, they abstract those behaviors).**
    - **But UML interface realization not generalization.**

OO Requirements

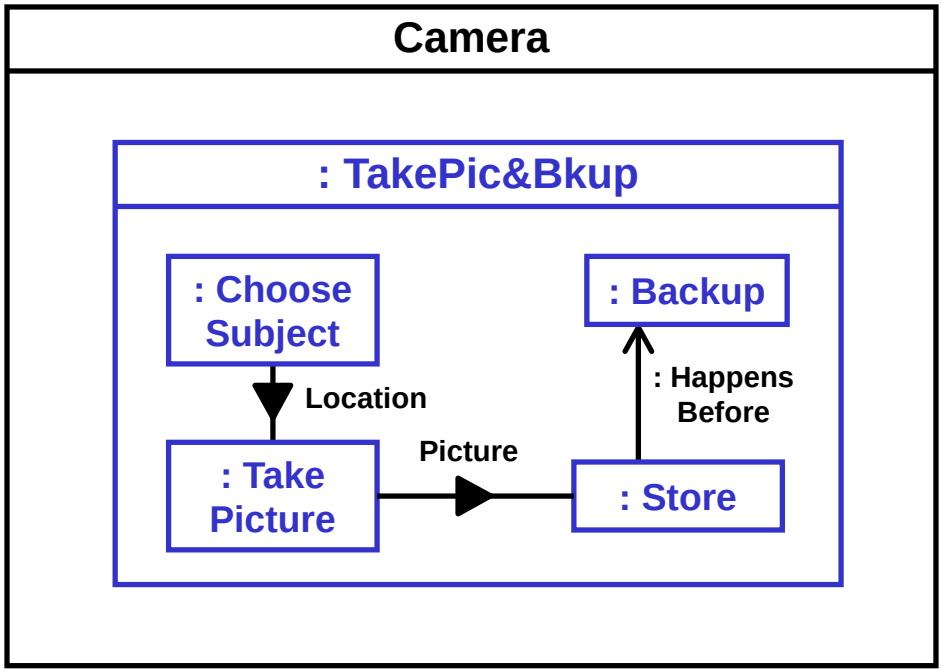
1. **Behavior encapsulation**
   - **“Surfaced” behaviors (no steps)**
2. **Behavior inheritance**
3. **Protocols**
   - **Expected order of using surfaced behaviors.**

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- **OO Behavior Solution**
  1. **Behavior encapsulation**
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary

Properties of Objects for Behavior Occurrences

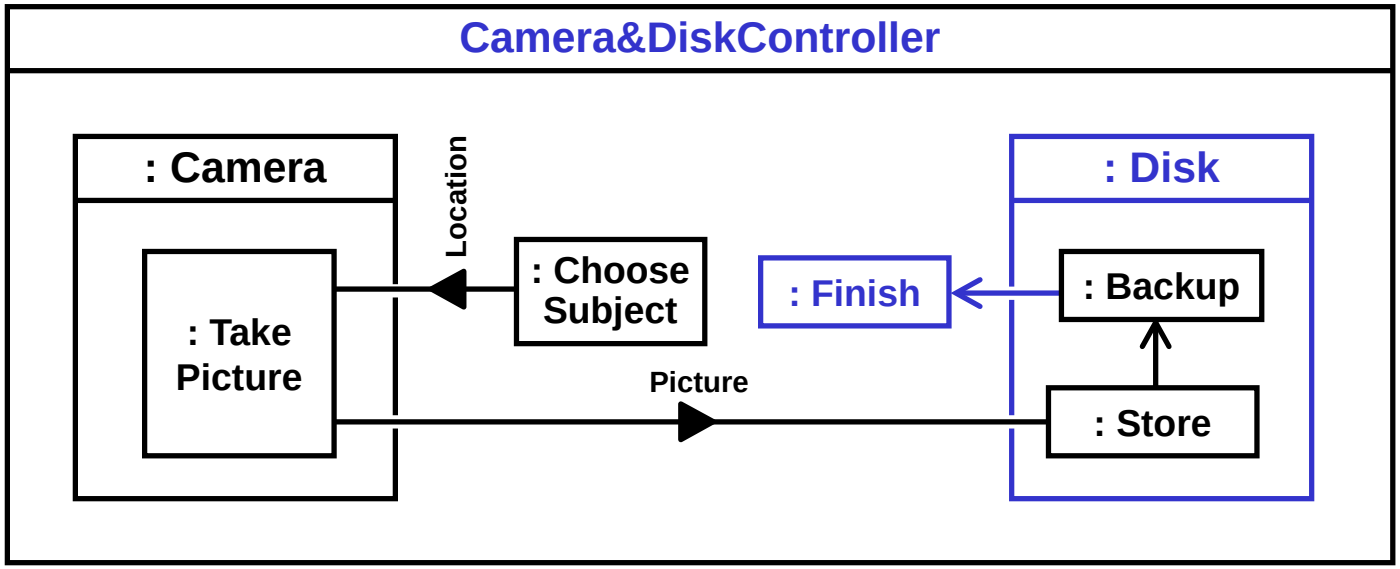
Model (M1)



- **Values of these properties are executions (occurrences, M0 instances) of behaviors.**
  - **For example, classifier behavior executions.**

Connecting Behavior Occurrence Properties Across Objects

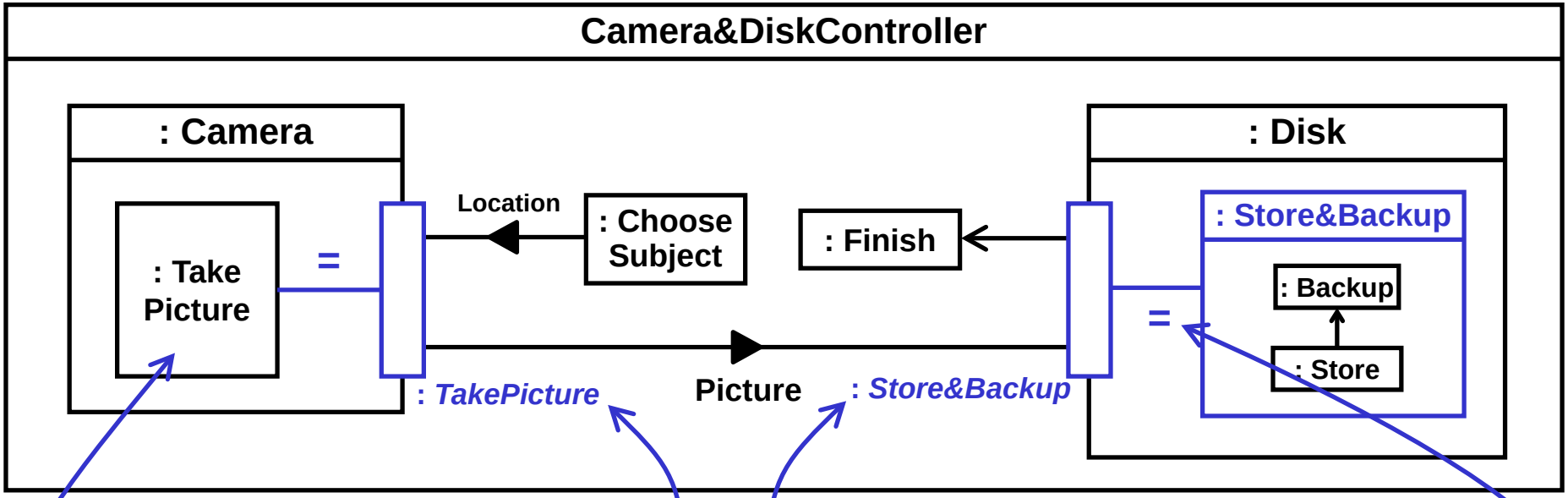
Model (M1)



- **Behaviors not encapsulated.**
  - **Controller specifies “how” picture is taken.**
  - **Compare to activity partitions.**
- **Controller should only specify inputs and outputs for camera and disk behaviors.**

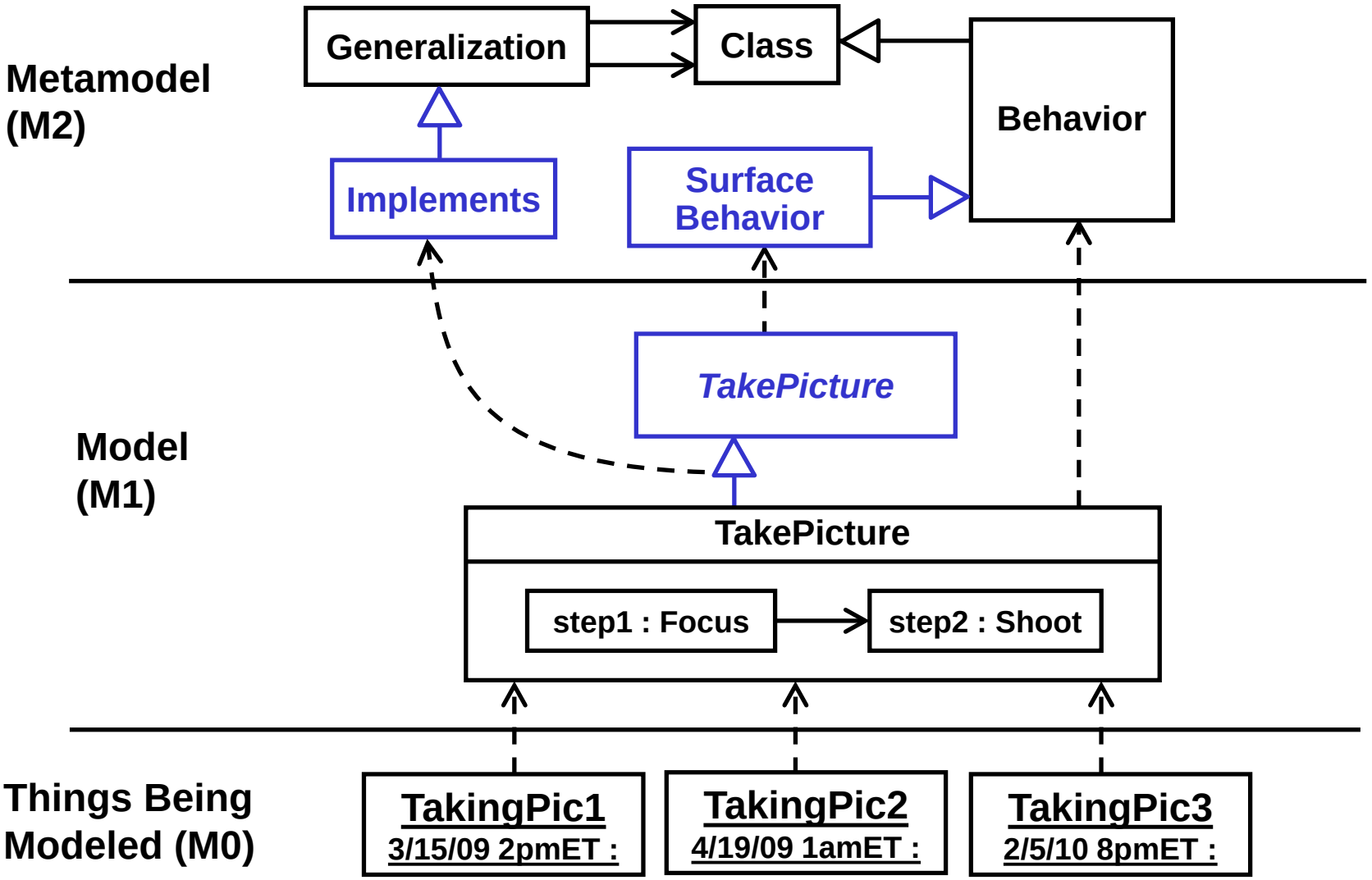
Encapsulating Behaviors

Model (M1)

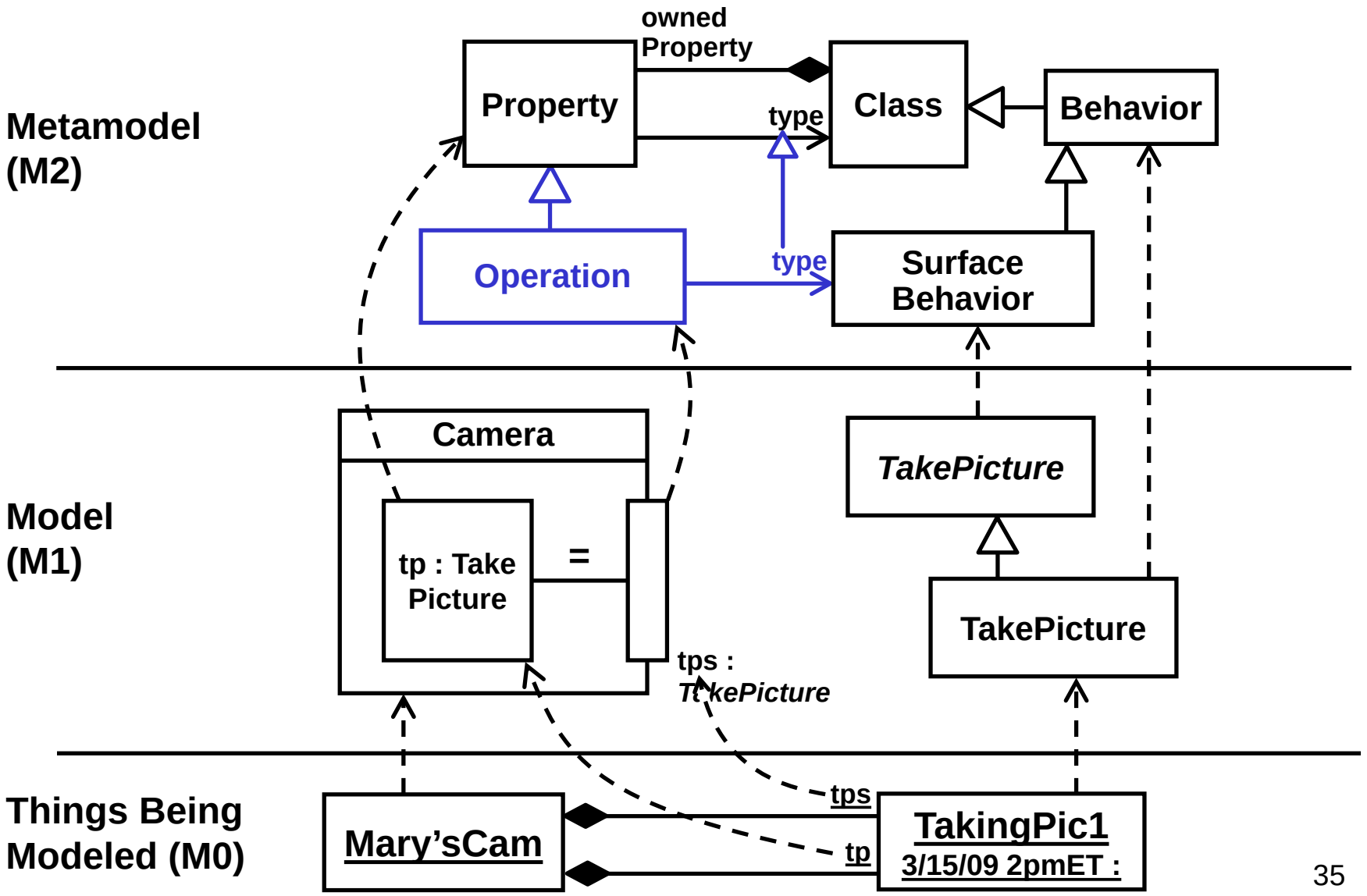


- **External behavior properties (operations)**
  - **Types only "expose" inputs and outputs.**
  - **Have same executions (equal values) as internal behavior properties (methods).**
- **(Not ports)**

External Behaviors

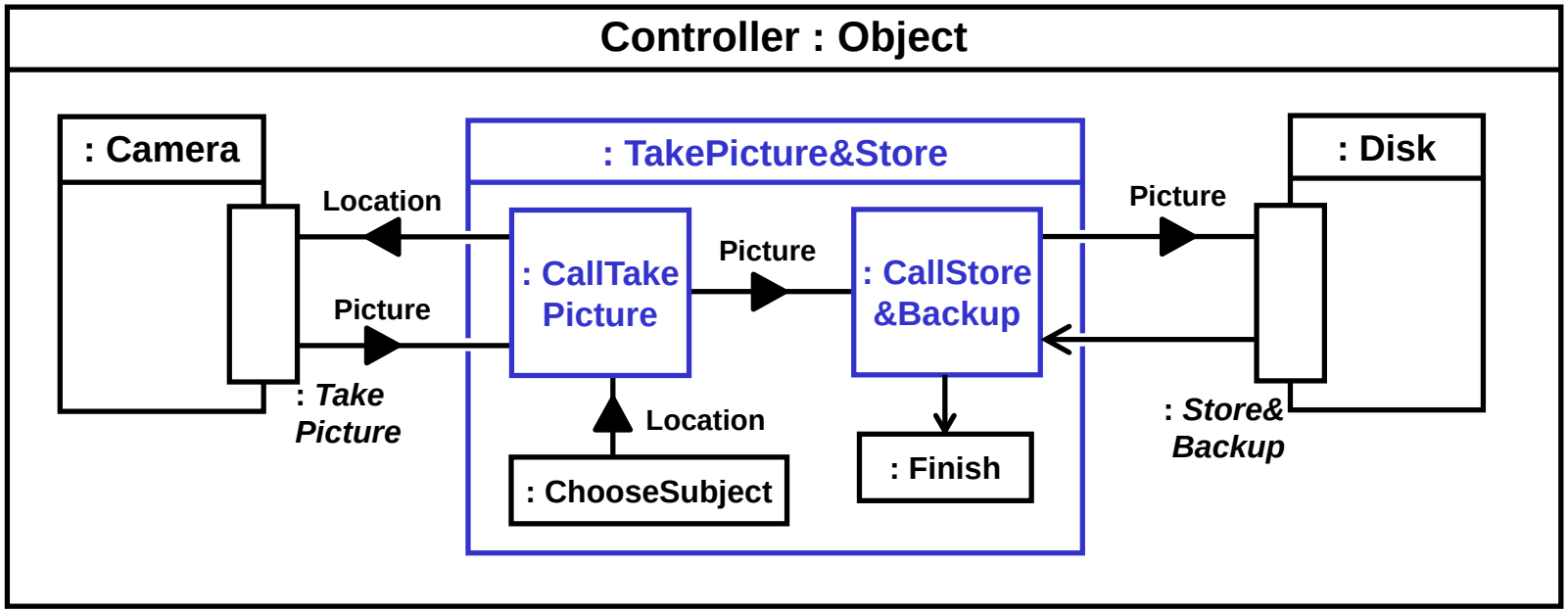


Operations


Metamodel
(M2)
Model
(M1)
Things Being
Modeled (M0)
owned
Property
Property
type
Class
Behavior
Operation
type
Surface
Behavior
Camera
tp : Take
Picture
=
tps :
TakePicture
TakePicture
TakePicture
tps
tp
Mary'sCam
TakingPic1
3/15/09 2pmET :

Behavior Invocation

Model (M1)

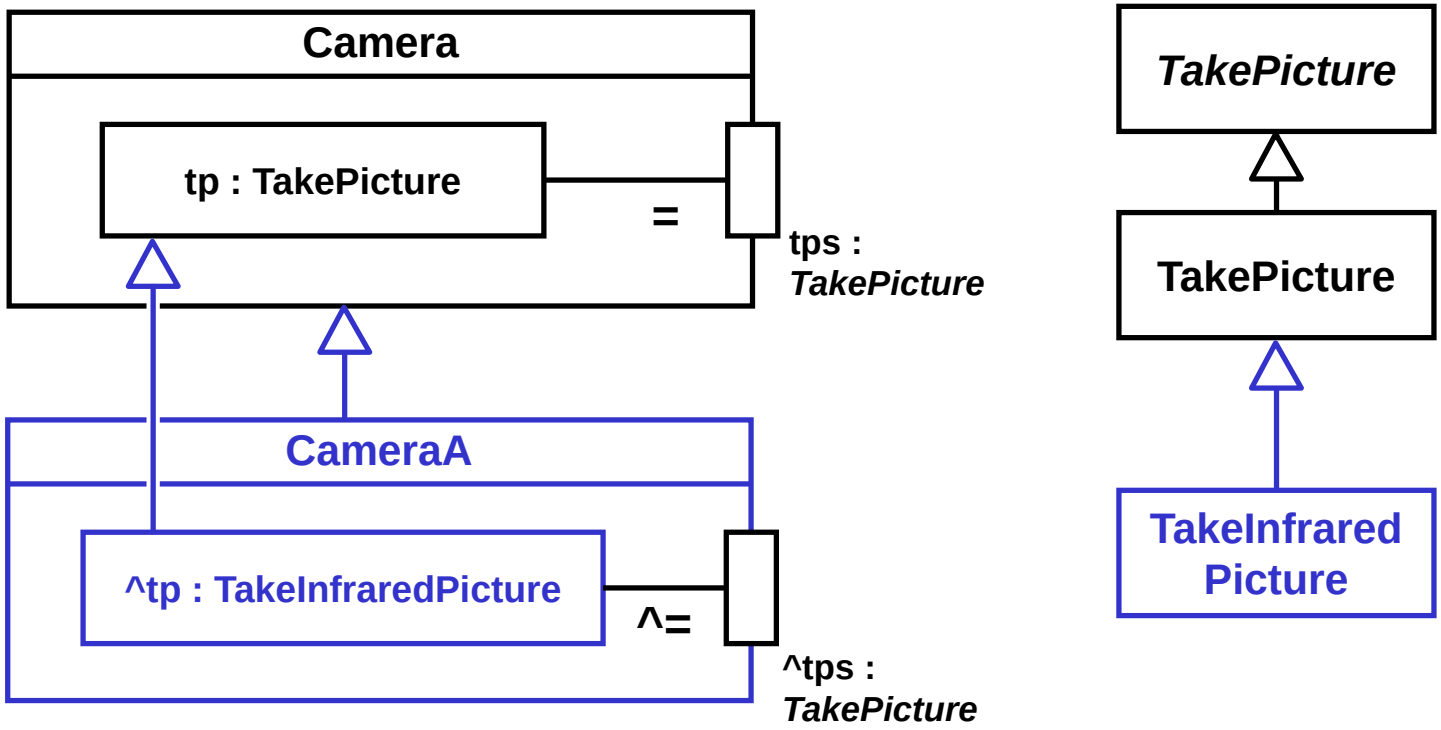


- **"Calls" are behaviors that constrain surrounding successions and item flows.**
  - **Specify whether to wait for return (synchronous/asynchronous calls).**
  - **Have no steps ("no-ops").**

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. **Behavior inheritance**
  3. Protocols (interaction and OO)
- Summary

Specializing Methods



- **Not OO "overriding":**
  - **Specialized methods cannot remove inherited elements, only specialize them.**
  - **Use general methods for commonality among implementations**

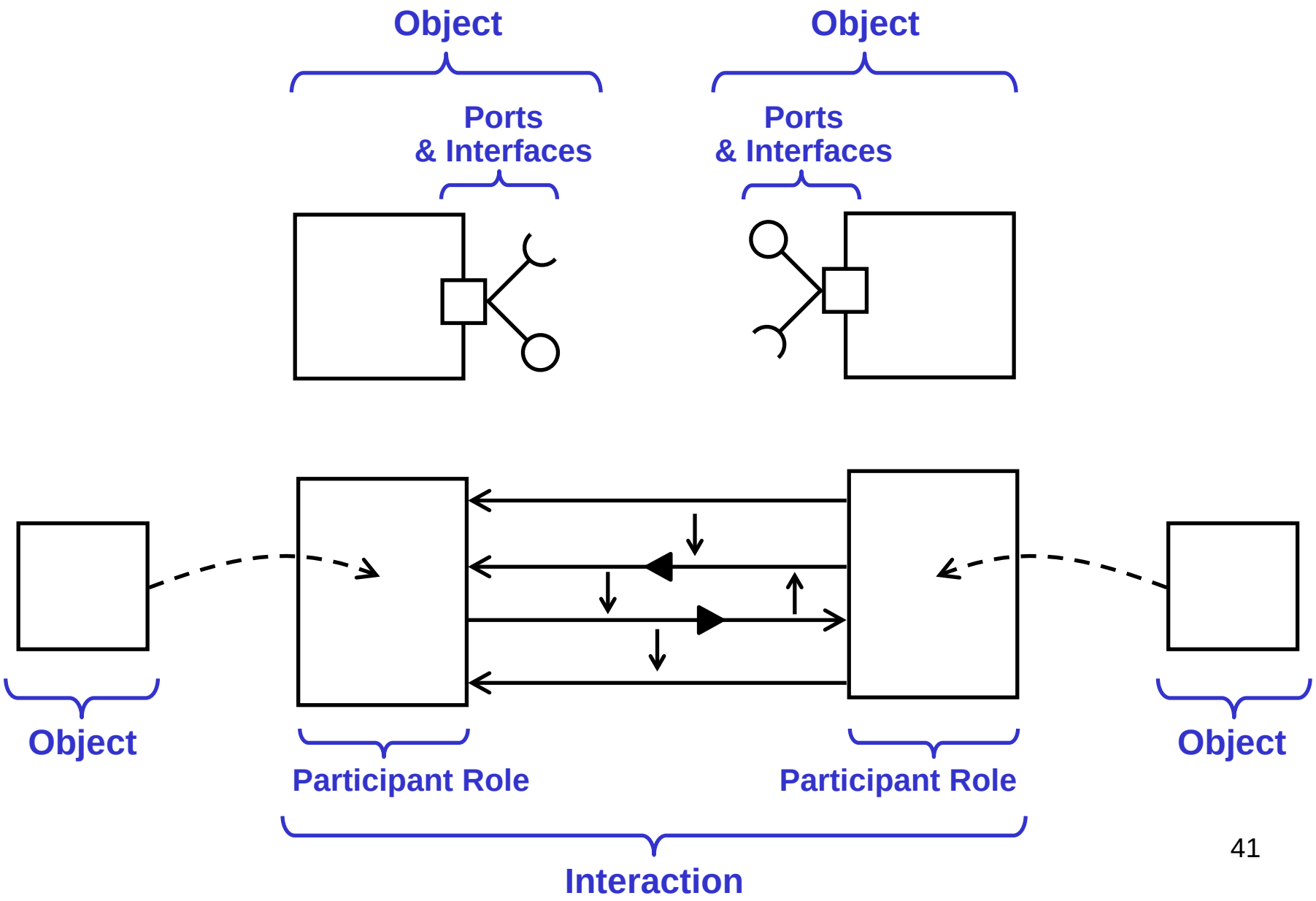
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. **Protocols (interaction and OO)**
- Summary

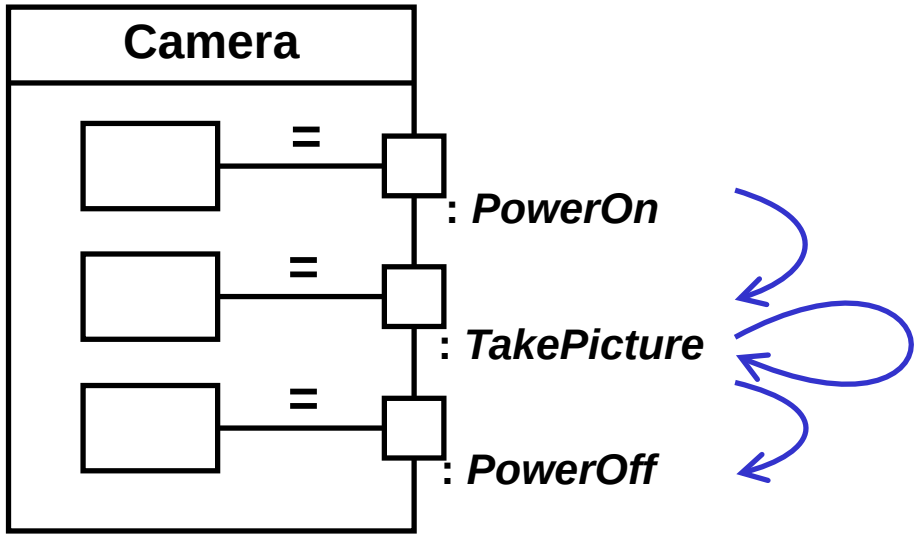
OO View of Interactions

- **Objects support interactions by providing “services” (including data).**
  - **UML added services required of other objects.**
- **Object models (classes) typically do not specify the interactions they support.**
  - **Only services “surfaced” to the outside.**
  - **Except for UML’s protocol state machines.**

OO & Interaction Approaches



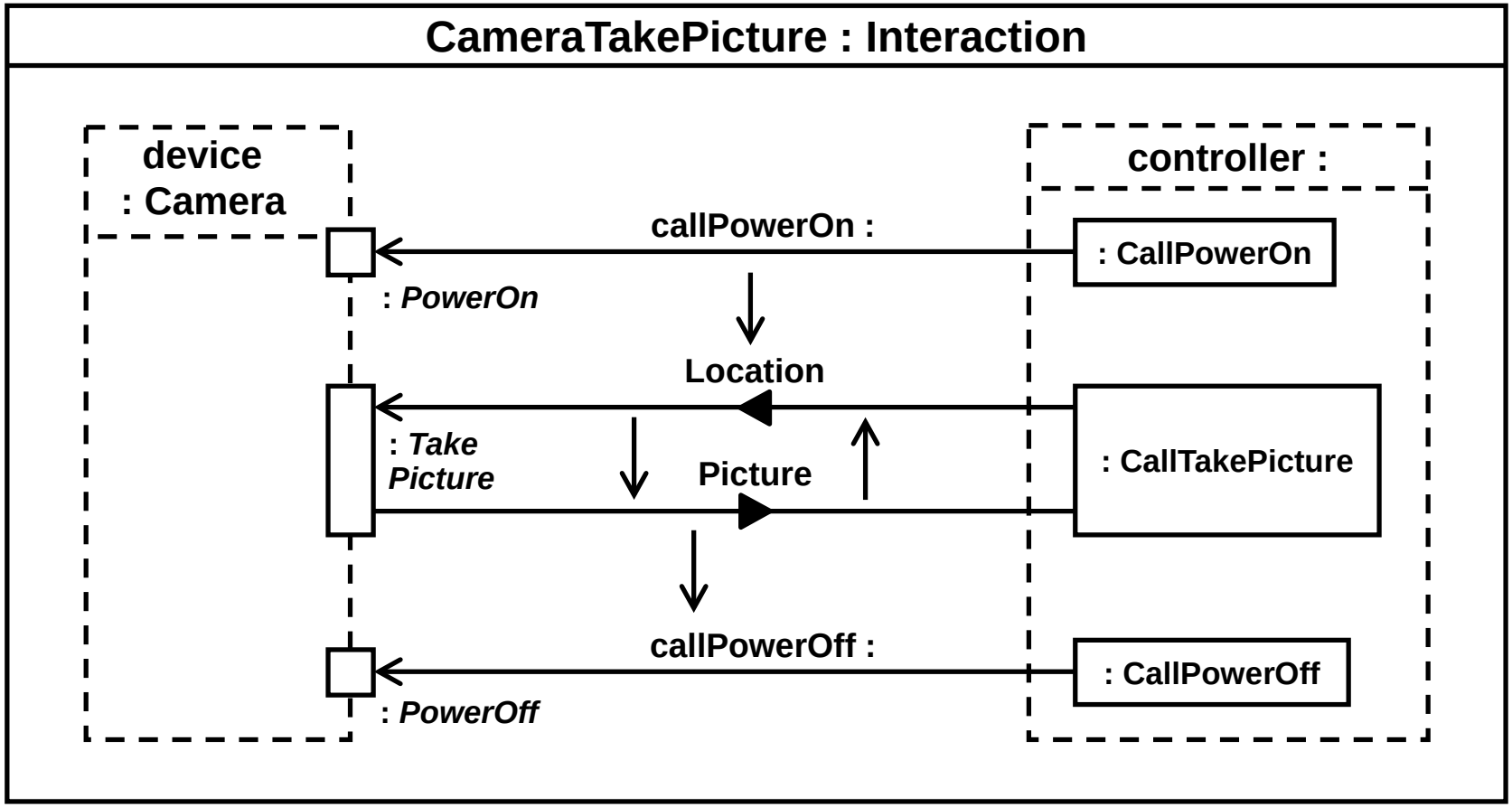
Protocols for Using Operations



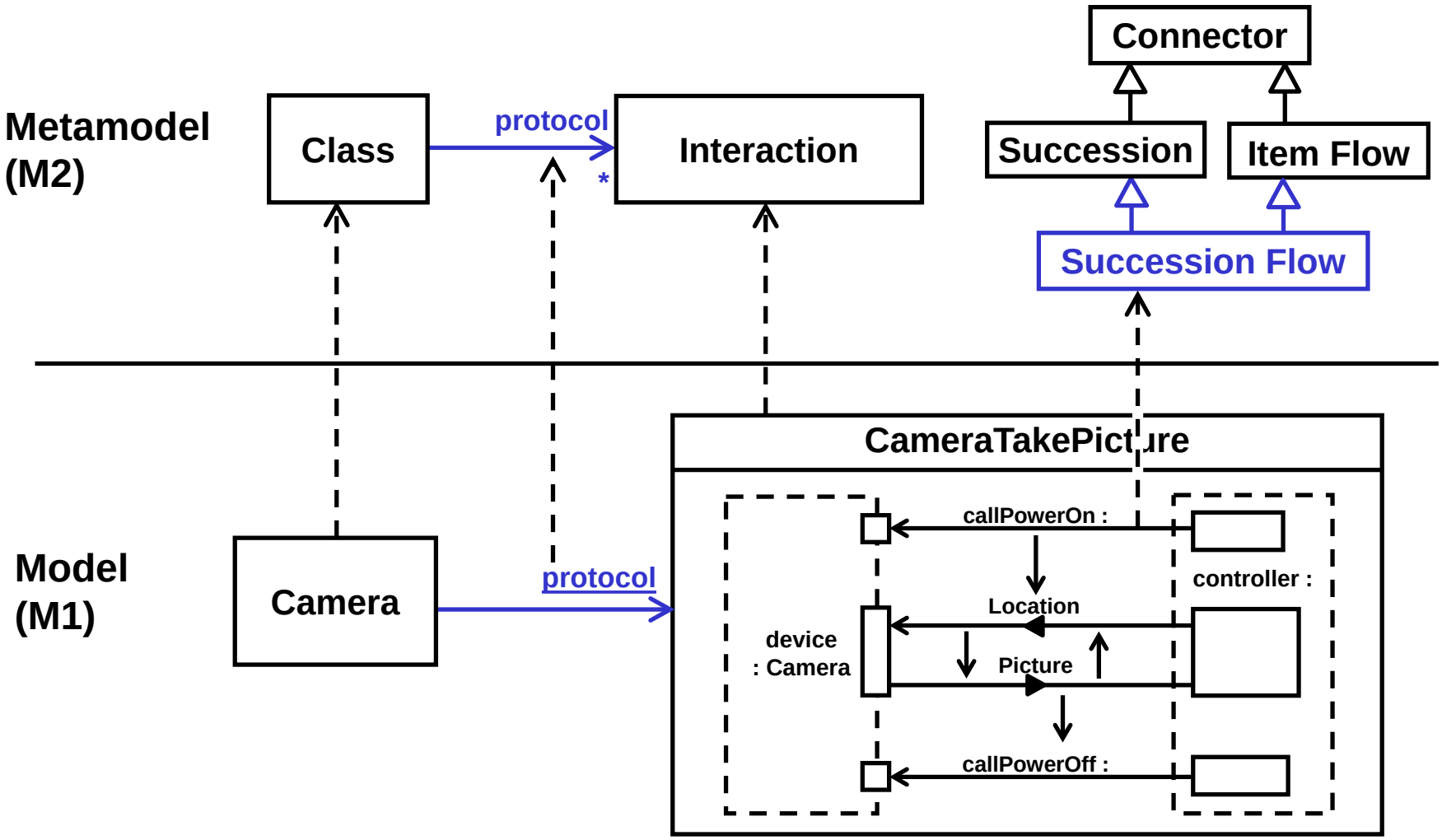
- **Protocol:**
  - **Power must be turned on before taking picture.**
  - **Multiple pictures can be taken.**
  - **Power must be turned off after the last picture is taken.**

Protocol as Interaction

Model (M1)

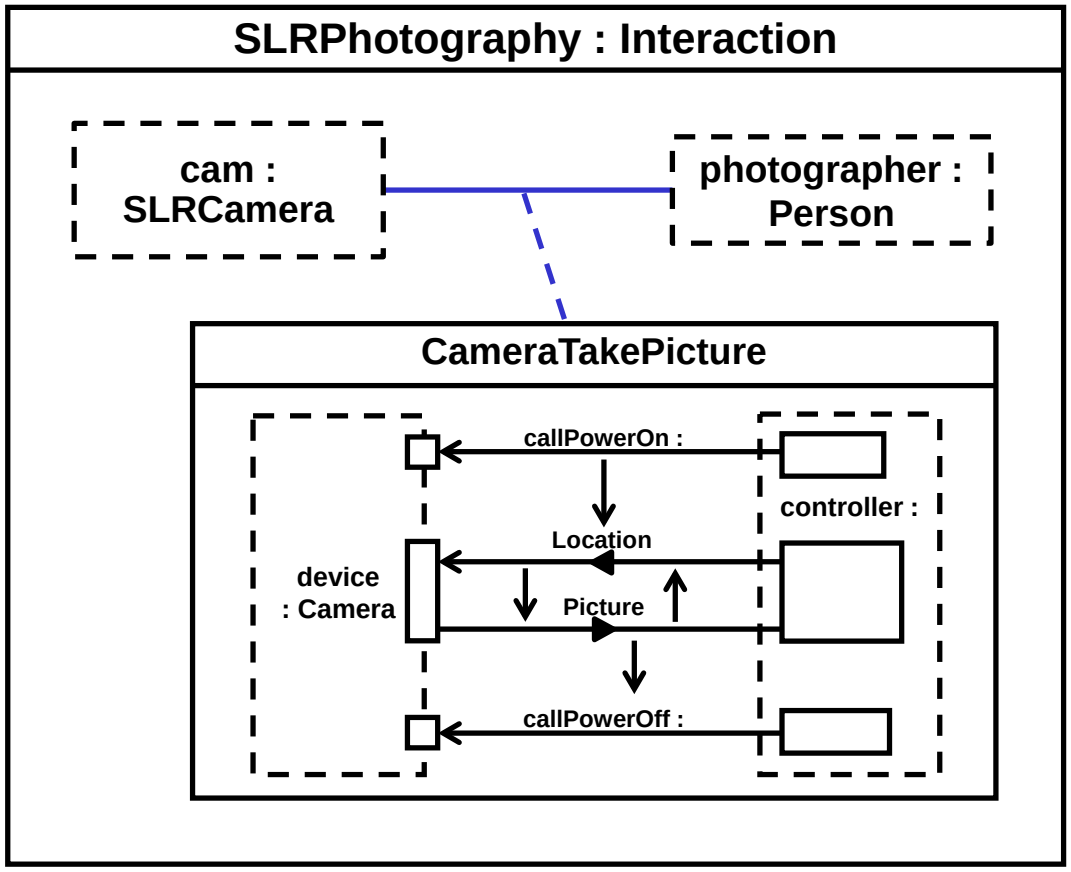


Protocol as Interaction (M2)



Using Interaction Protocols

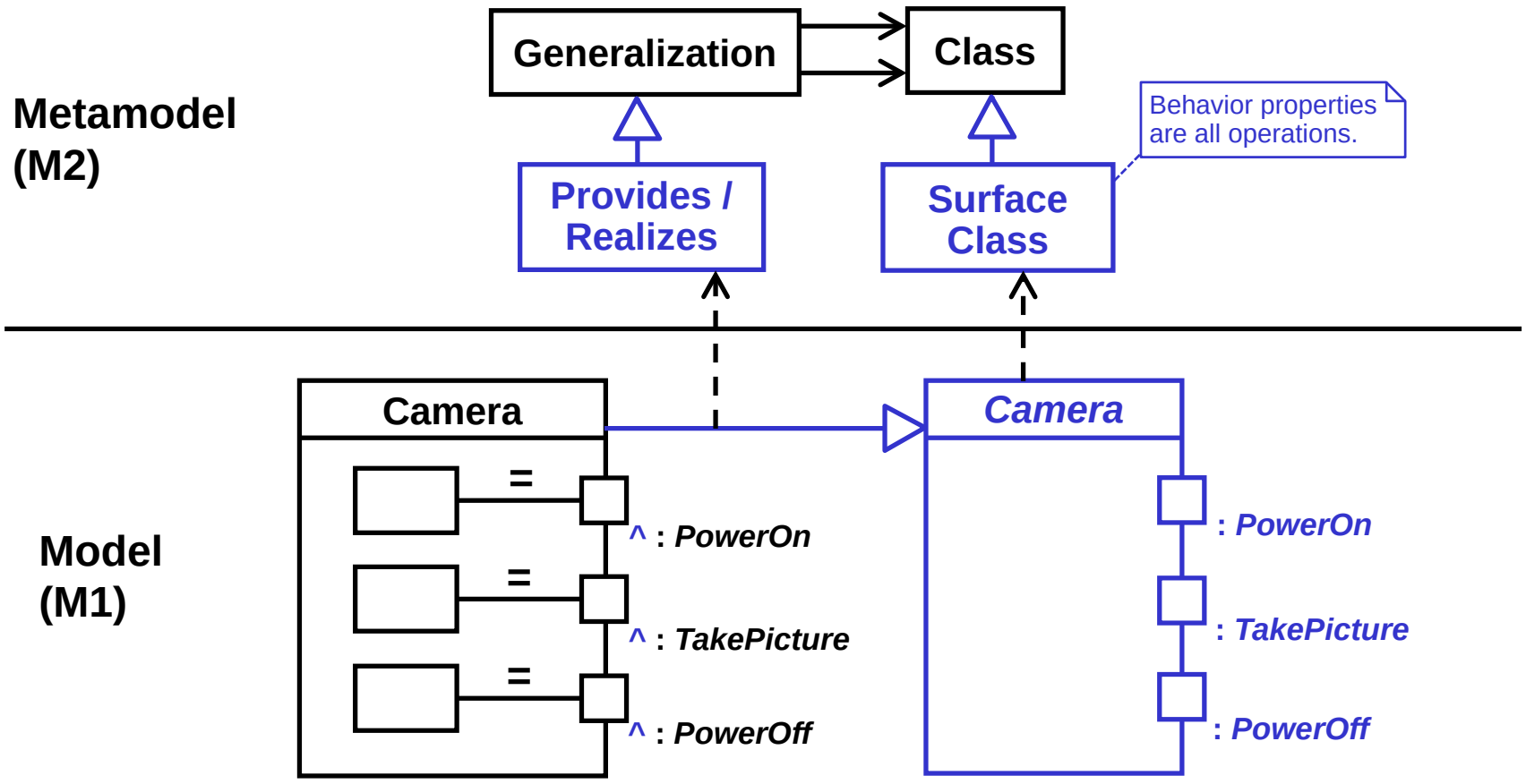
Model (M1)



Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. **Protocols (interaction and OO)**
- Summary

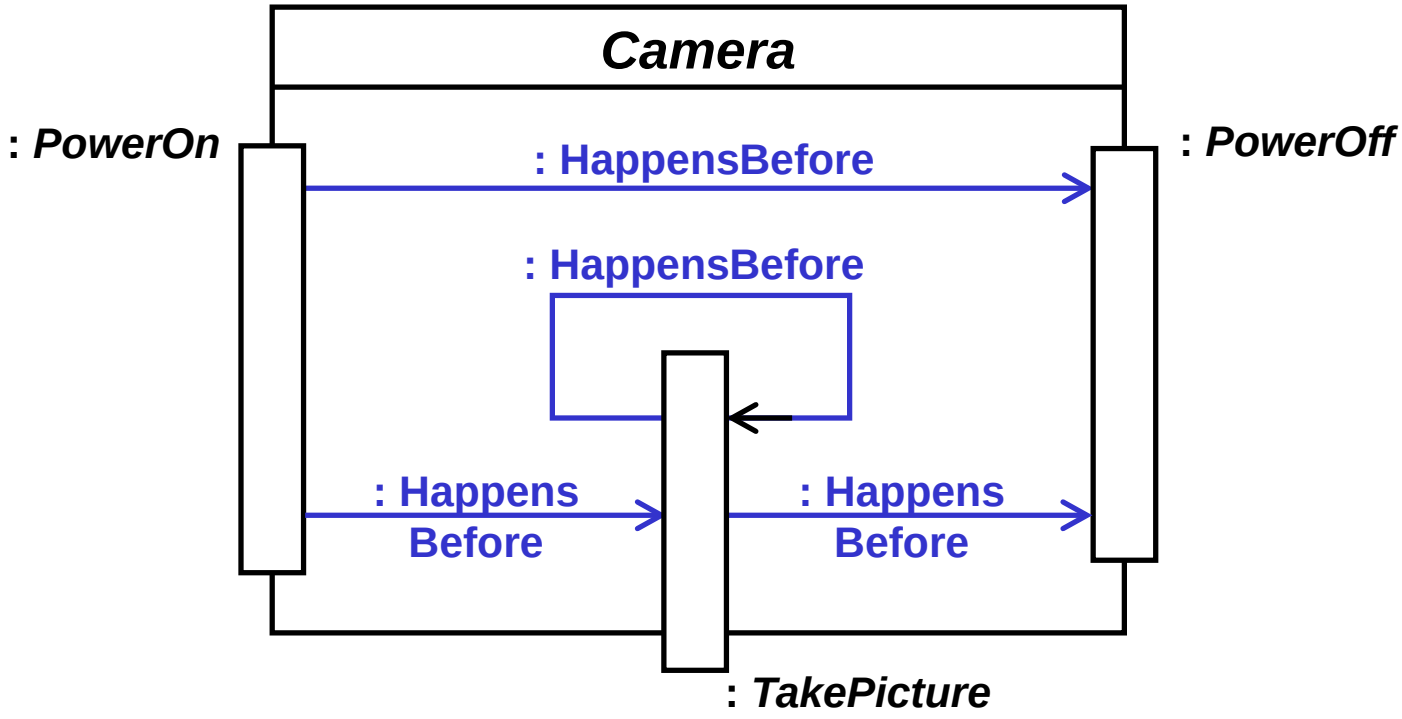
OO Interfaces



- **Could use as interaction participant types.**

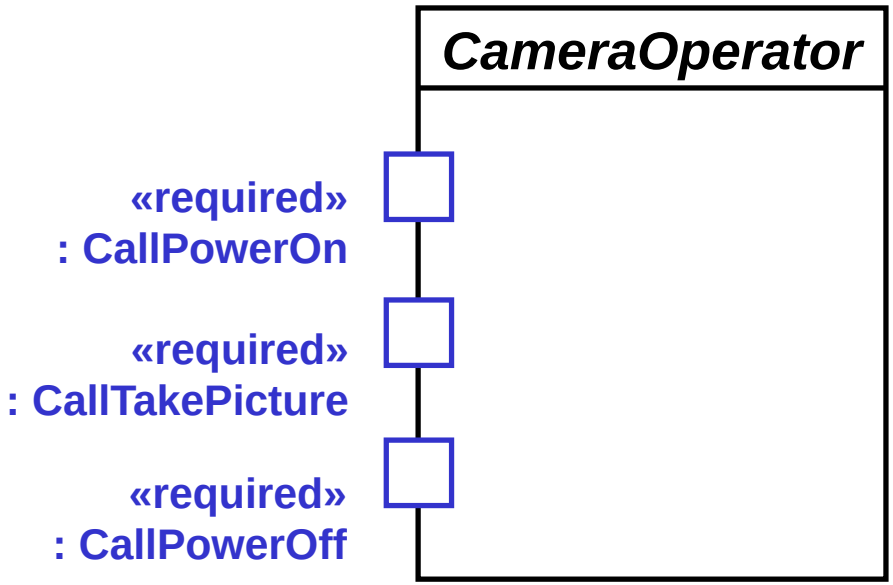
OO Protocols

Model (M1)



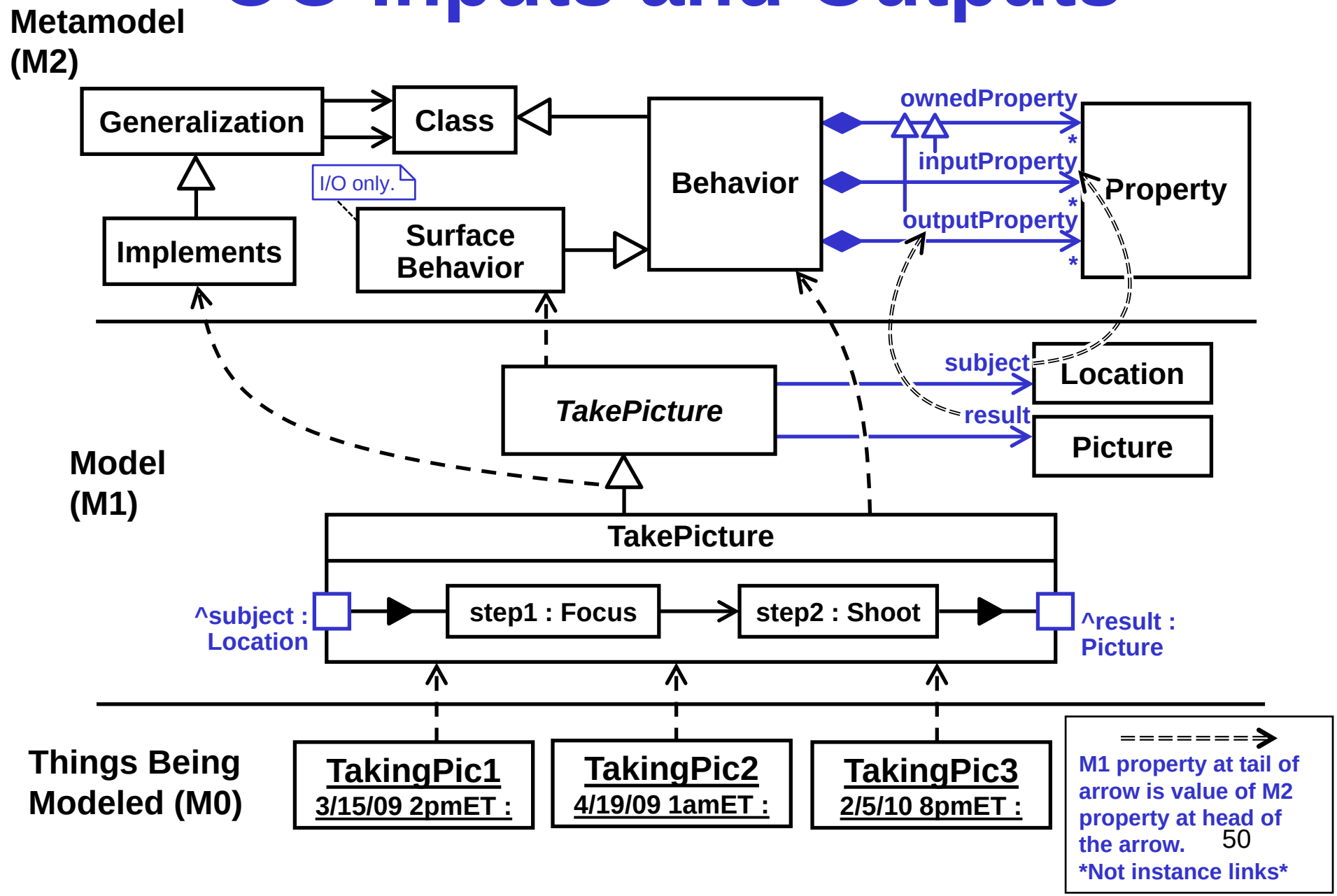
- **Defined without external objects.**

Conjugation

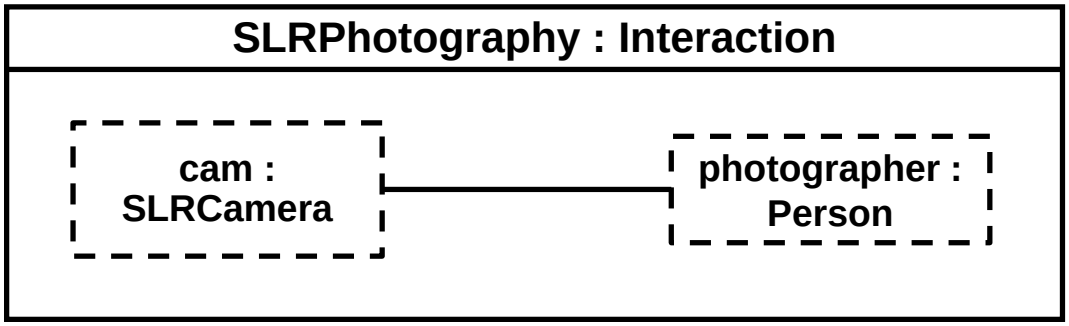


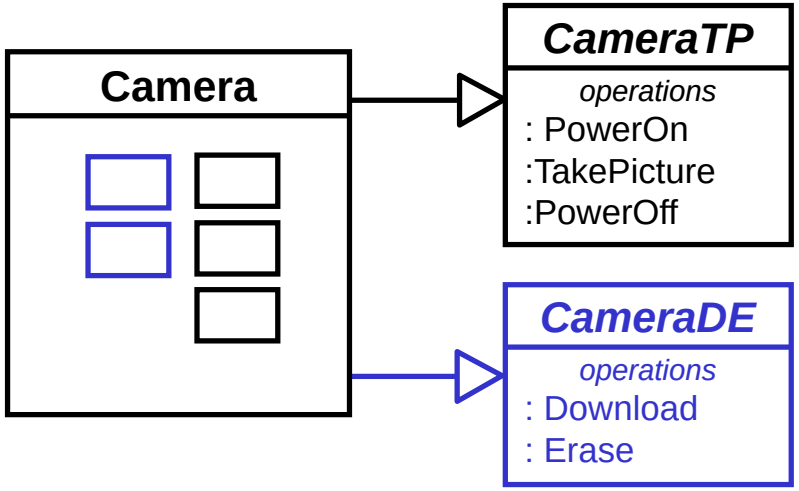
- **UML required operations = service requests sent to external objects.**

OO Inputs and Outputs



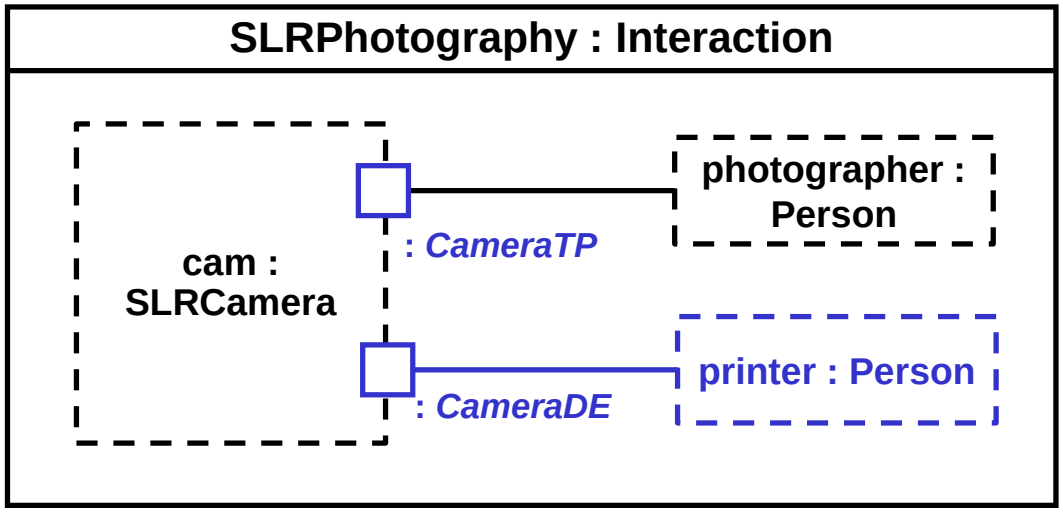
Multiple OO Interfaces





- **Connector uses both interfaces or one?**
  - **If one, which?**

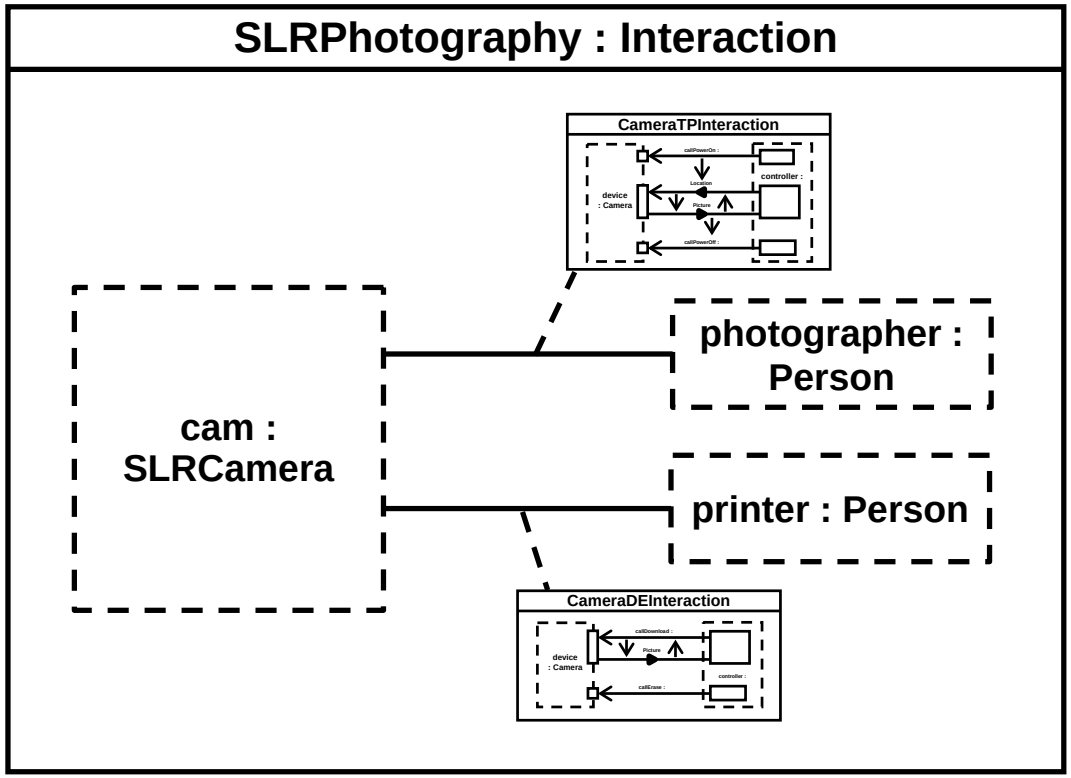
Port for Each OO Interface



- **Typed by interfaces, not operations.**
- **Raises questions:**
  - **Are ports separate from objects they're on?**
  - **If separate, are they internal or external parts?**
  - **Tied up an entire SysML RTF.**

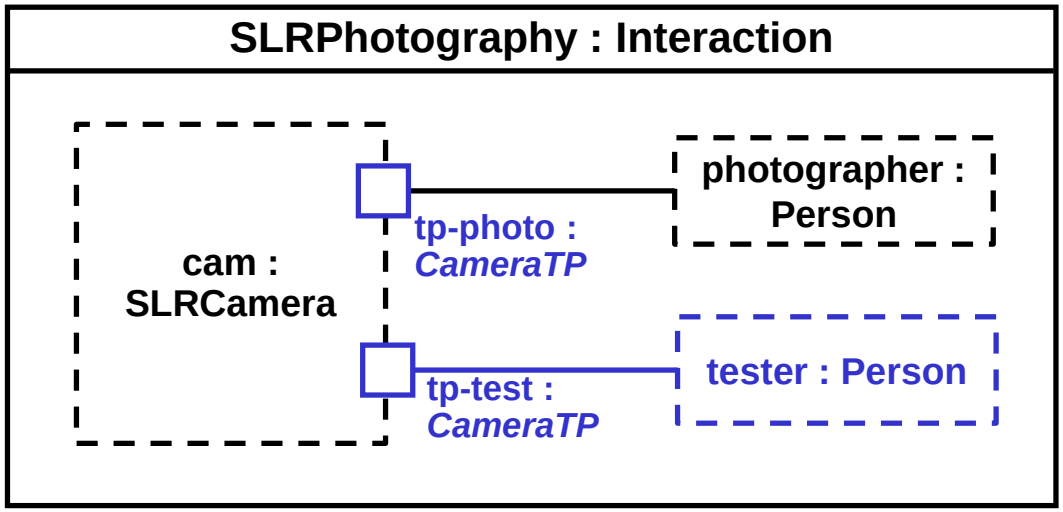
Multiple Interaction Protocols

Model (M1)



- **Connectors typed by different interactions.**
  - **Ports not needed.**

Multiple Ports for Same Interface



- **Object can interact differently based on port used.**
  - **Better to define with separate interactions.**
- **If same interaction, use correlation (BPMN).**
- **Not possible with interaction protocols.**

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- **Summary**

Summary

- **Unify OO behavior using**
  - **Properties for operations and methods**
  - **Inheritance for “overriding” methods.**
- **Simplify protocol modeling with**
  - **Interactions instead of OO interfaces & ports.**
- **Speeds learning and analysis integration.**

More Information

- **Intro to Behavior as Composite Structure**
  - http://doc.omg.org/ad/2018-03-02
- **Interaction as Composite Structure**
  - http://doc.omg.org/ad/18-06-11
- **Additional slides**
  - **Starts with onto, includes interactions.**
  - http://conradbock.org/bock-ontological-behavior-modeling-jpl-slides.pdf
- **Paper:** http://dx.doi.org/10.5381/jot.2011.10.1.a3
- **Application to BPMN:** http://conradbock.org/#BPDM
- **KerML: Contact Chas Galey** charles.e.galey@lmco.com