Logical Behavior Modeling for UML:

Behavior as Composite Structure

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

James Odell, Thematix

Tim Weilkiens, oose Innovative Informatik

James Baker, Sparx Systems

Antoine Lonjon, MEGA

September 13, 2009
(revised July 16, 2010 for DODAF DM2 WG,
Dec 31, 2013 for JPL)

Overview

- Motivation
- Logical modeling
- Composite structure
- Behaviors as composites
  - Sequencing
    - Aside on SysML extensions for logical modeling
  - Events
  - Participants
  - Flows (object flows and messaging)
    - External participants
    - Flow ordering
    - Composition with flows and participants

Motivation

- UML has three behavior diagrams.
  - Activity, state, interaction.
- Three underlying metamodels.
- Very little integration.
- Develop an integrated behavior metamodel for the three notations.
  - Implies focus on meaning rather than notation.

Logical Modeling

- Motivation
- **Logical modeling**
- Composite structure
- Behaviors as composites
  - Sequencing
    - Aside on SysML extensions for logical modeling
  - Events
  - Participants
  - Flows (object flows and messaging)
    - External participants
    - Flow ordering
    - Composition with flows and participants

Quantitative Modeling

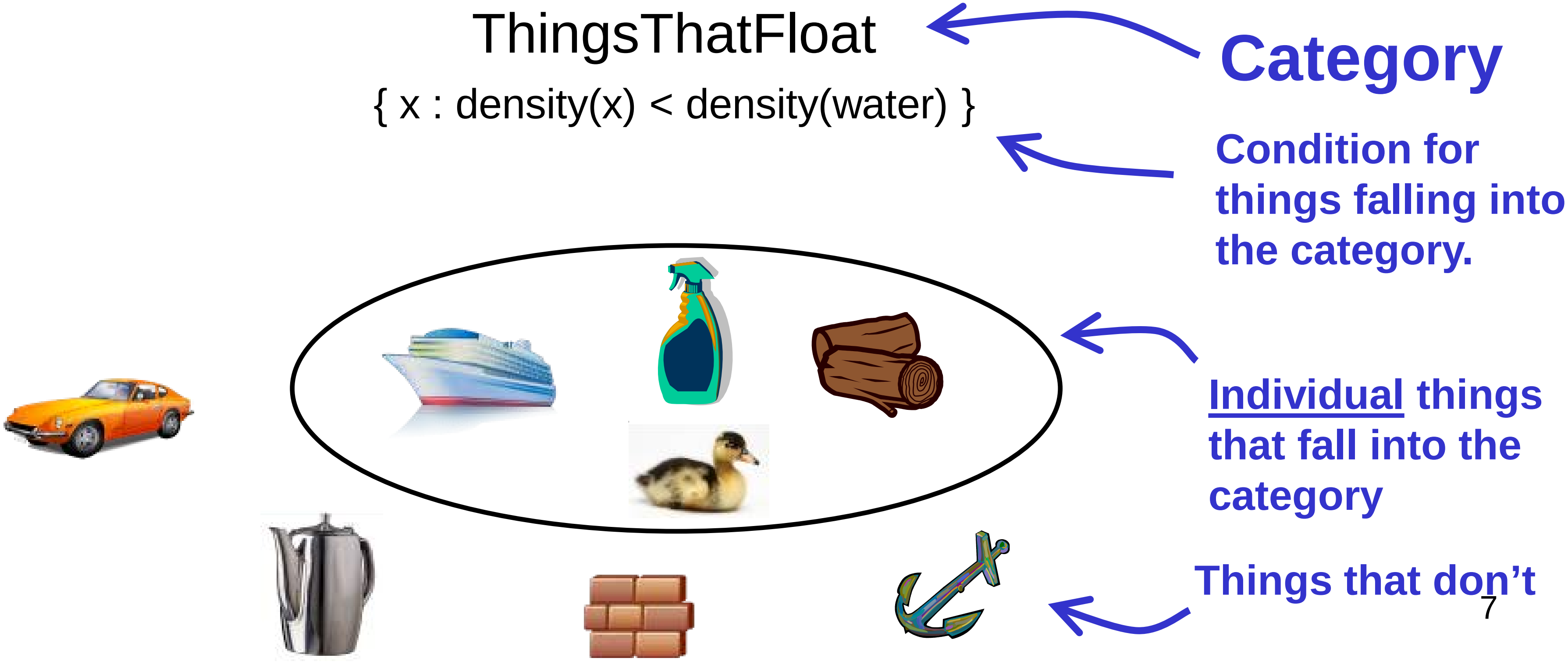
- Quantitative modeling
  - Numerical formulas (equations)
  - Dynamic and stochastic simulations
- Used for:
  - Calculating or simulating numeric values and probabilities.
  - Deriving new numerical formulas.

Logical Modeling

- Logical modeling is about categorizing things and relations between things ...
  - This document is a requirement, this other one is a design, and the second satisfies the first.
- … and keeping these categorizations consistent.
  - Requirements or designs are changed, does the satisfies relation still hold?
  - If not, what would make it hold again?

Categories = Conditions for Things in the Category

- Things “fall into” categories.
- Categories have conditions for what can and cannot fall into the category.



ThingsThatFloat

{ x : density(x) < density(water) }

Category

Condition for things falling into the category.

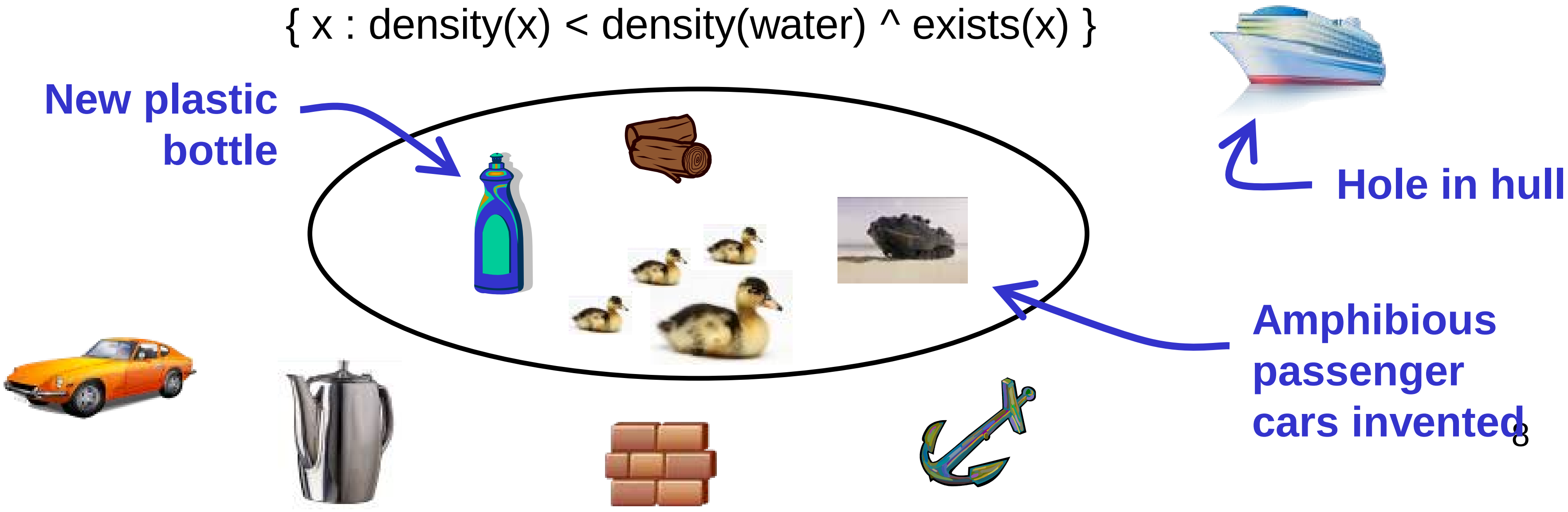
Individual things that fall into the category

Things that don’t

Categories only Specify Sets, they are not sets themselves

- Which things fall into a category can change over time without changing the category (condition).
  - New things created, some things destroyed, conditions met or unmet over time.
  - Not true for set membership.

ThingsThatFloat



{ x : density(x) < density(water) ^ exists(x) }

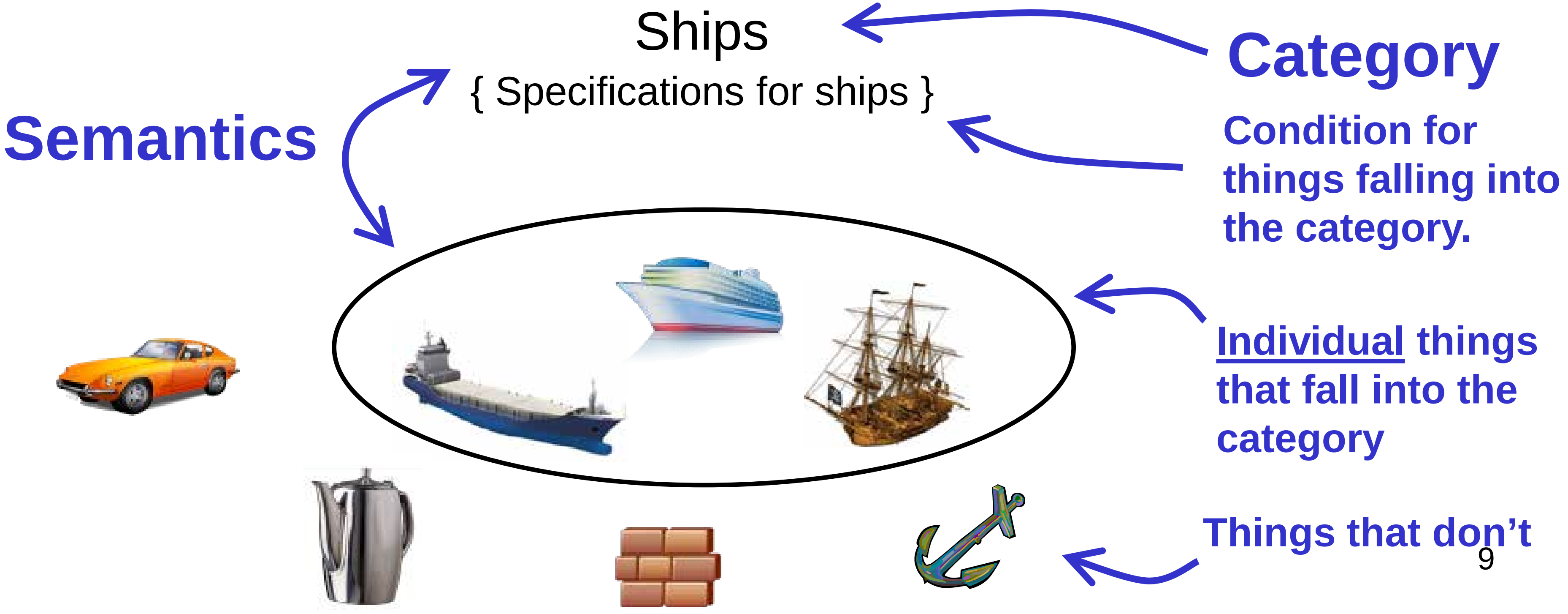
New plastic bottle

Hole in hull

Amphibious passenger cars invented

Applied to Language Semantics

- **Modeling language semantics = How systems will be and behave when they are built according to a user model.**



Ships

{ Specifications for ships }

Semantics

Category

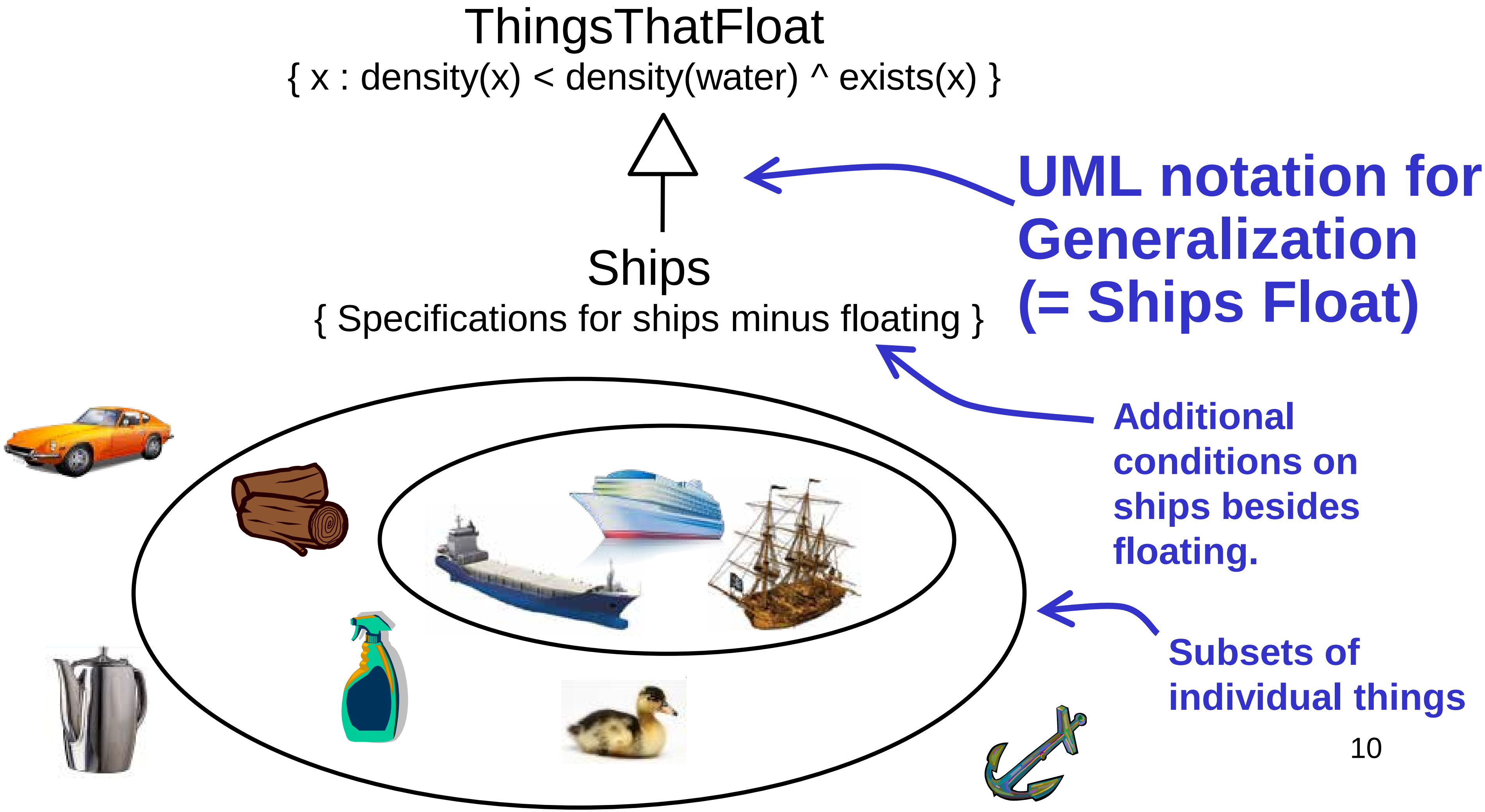
Condition for things falling into the category.

<u>Individual</u> things that fall into the category

Things that don’t

Category Generalization

- = Everything in one category is in another.



ThingsThatFloat

{ x : density(x) < density(water) ^ exists(x) }

Ships

{ Specifications for ships minus floating }

UML notation for Generalization (= Ships Float)

Additional conditions on ships besides floating.

Subsets of individual things

Formal Languages for Categorization

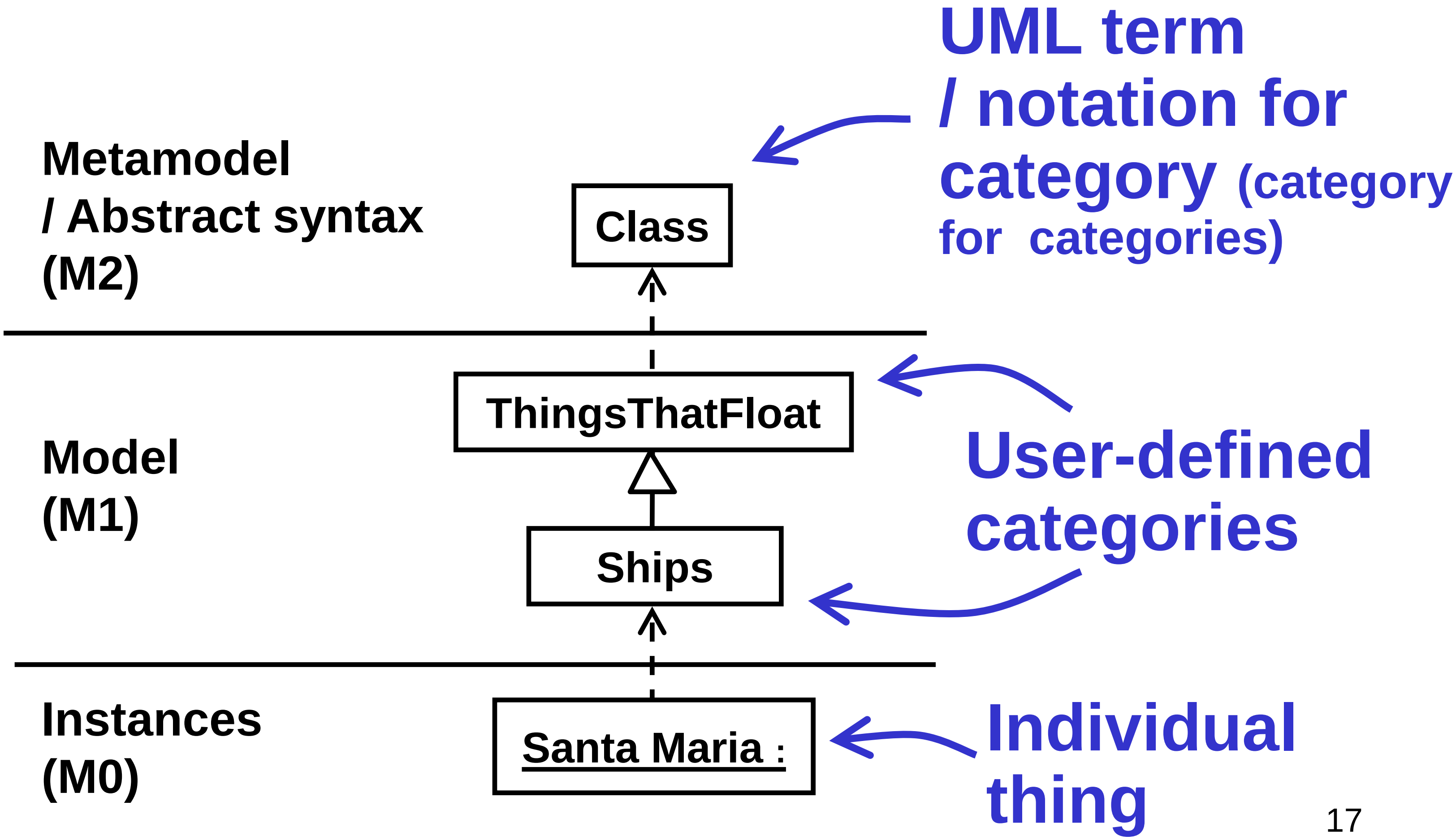
- First order logic and some of its specializations (“fragments”).
- Description Logic / Ontology Web Language (OWL / SROIQ DL).
- Model-theoretic semantics (formally relating categories and things falling into them).
- Widely supported by efficient automated reasoners.

Informal Languages for Categorization

- Many of these.
- Unified Modeling Language and its extensions.
  - Categorization semantics added in UML 2, alongside object-orientation.
  - Specified in free text, for example:
    - “An instance of a Classifier is also an (indirect) instance of each of its generalizations.”
- OWL/DL has been applied to formalize UML semantics.
  - OWL 2 has a specialization for UML-like languages.

Categorization in UML

- Repeated categorization in UML = metalevels.



Metamodel / Abstract syntax (M2)

Class

UML term / notation for category (category for categories)

Model (M1)

ThingsThatFloat

Ships

User-defined categories

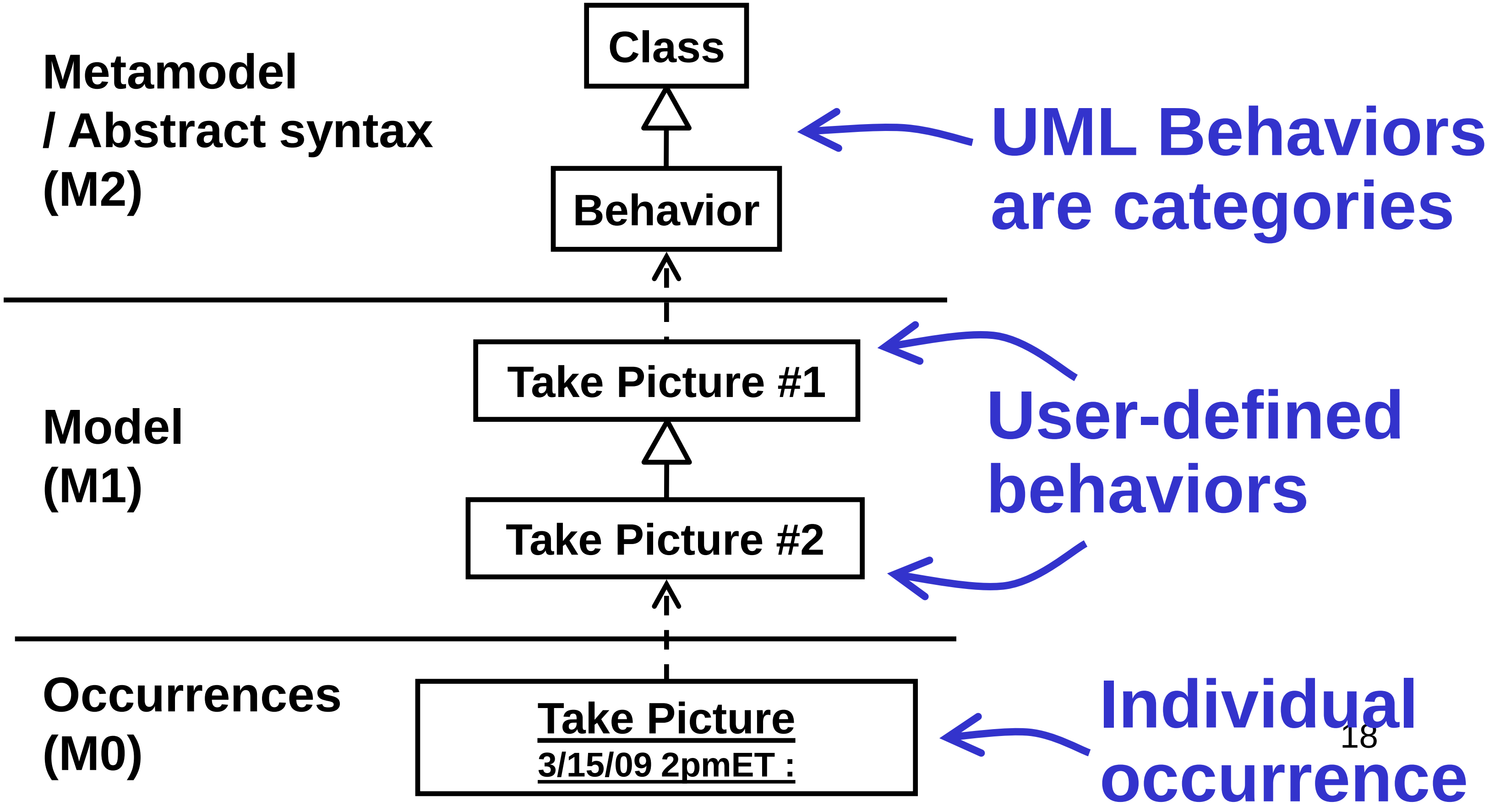
Instances (M0)

<u>Santa Maria :</u>

Individual thing

Categorization applied to UML Behaviors

- Behaviors are
  - Classes (modeled by M2 generalization).
  - specialized at M1 by user.
  - *occur* (execute, are performed) at M0.



Metamodel
/ Abstract syntax
(M2)

Class

Behavior

UML Behaviors
are categories

Model
(M1)

Take Picture #1

Take Picture #2

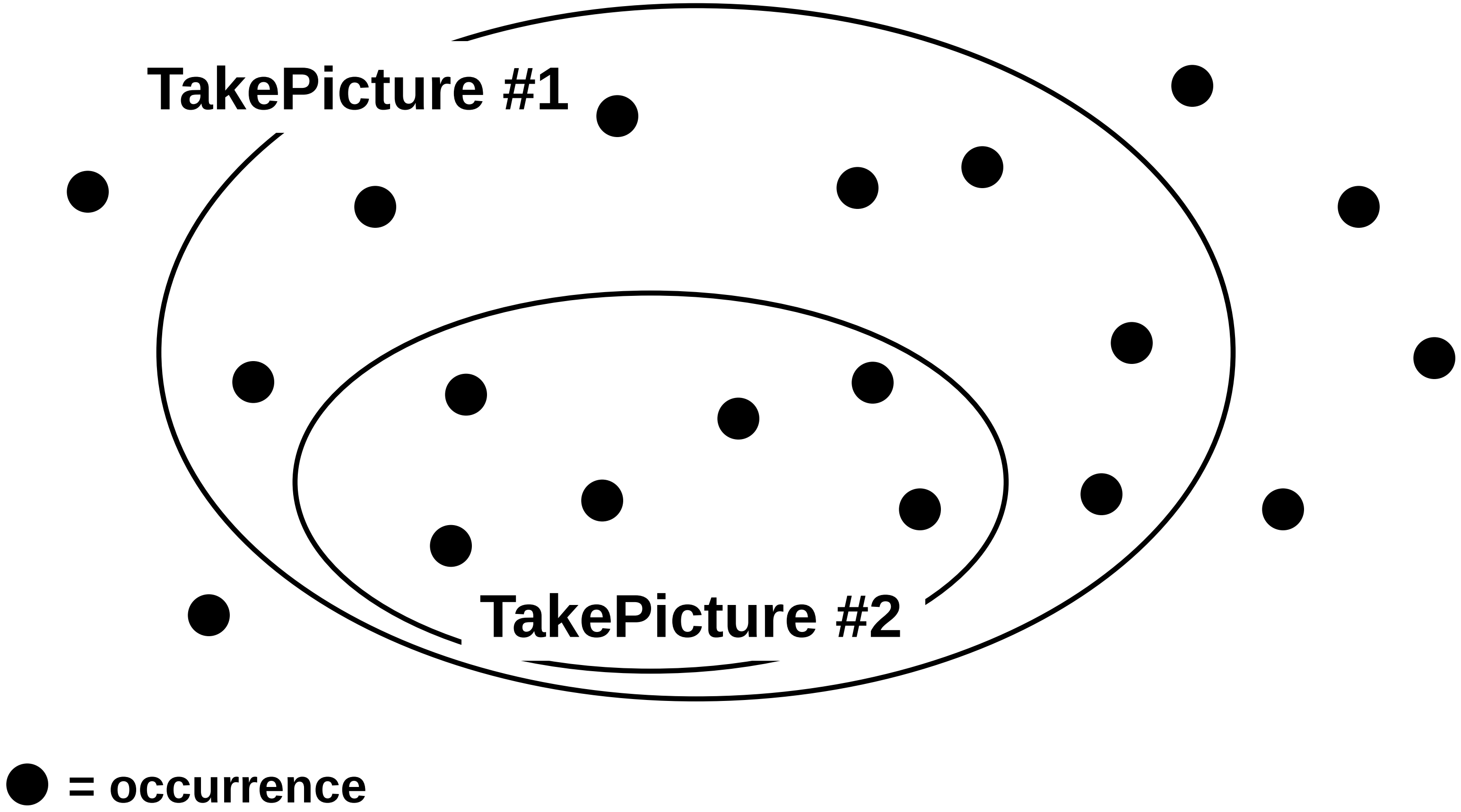
User-defined
behaviors

Occurrences
(M0)

Take Picture
3/15/09 2pmET :

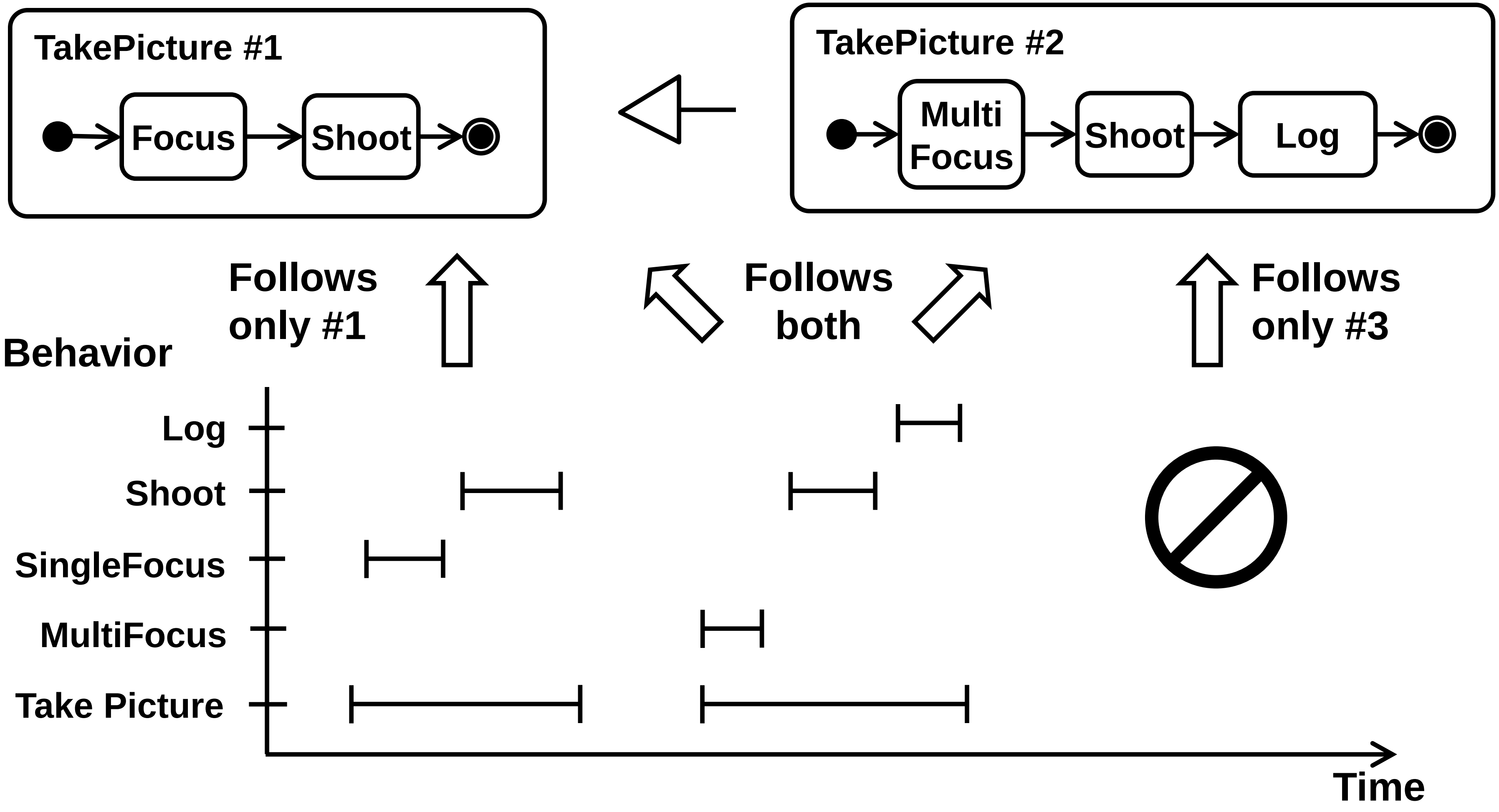
Individual
occurrence

Behavior Generalization



- Venn diagram illustration of previous example.

Behavior Generalization



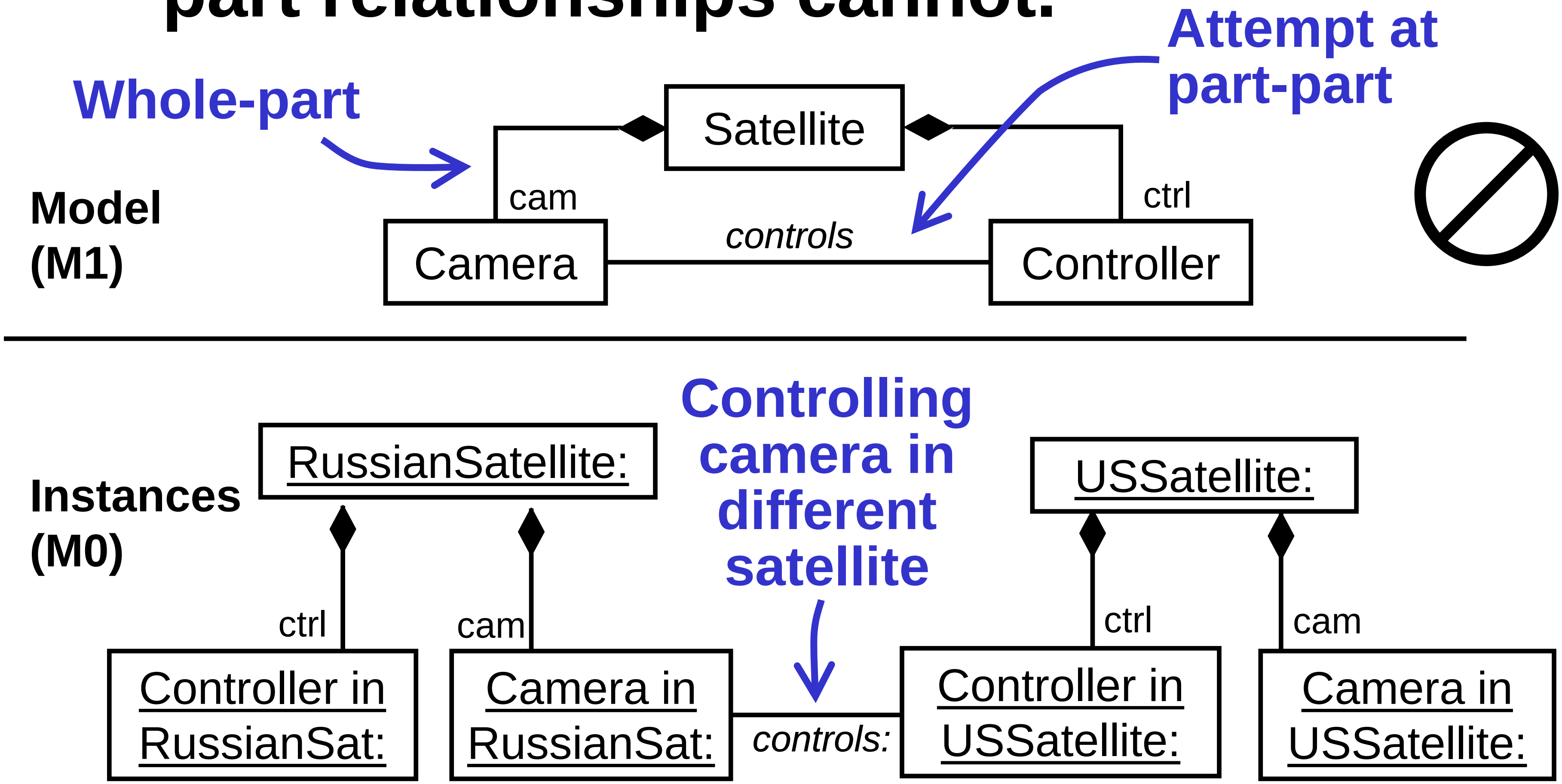
- **By the definition of generalization:**
  - Every occurrence (instance) of the specialized behavior (class) is an occurrence of the general behavior.

Composite Structure

- Motivation
- Logical modeling
- **Composite structure**
- Behaviors as composites
  - Sequencing
    - Aside on SysML extensions for logical modeling
  - Events
  - Participants
  - Flows (object flows and messaging)
    - External participants
    - Flow ordering
    - Composition with flows and participants

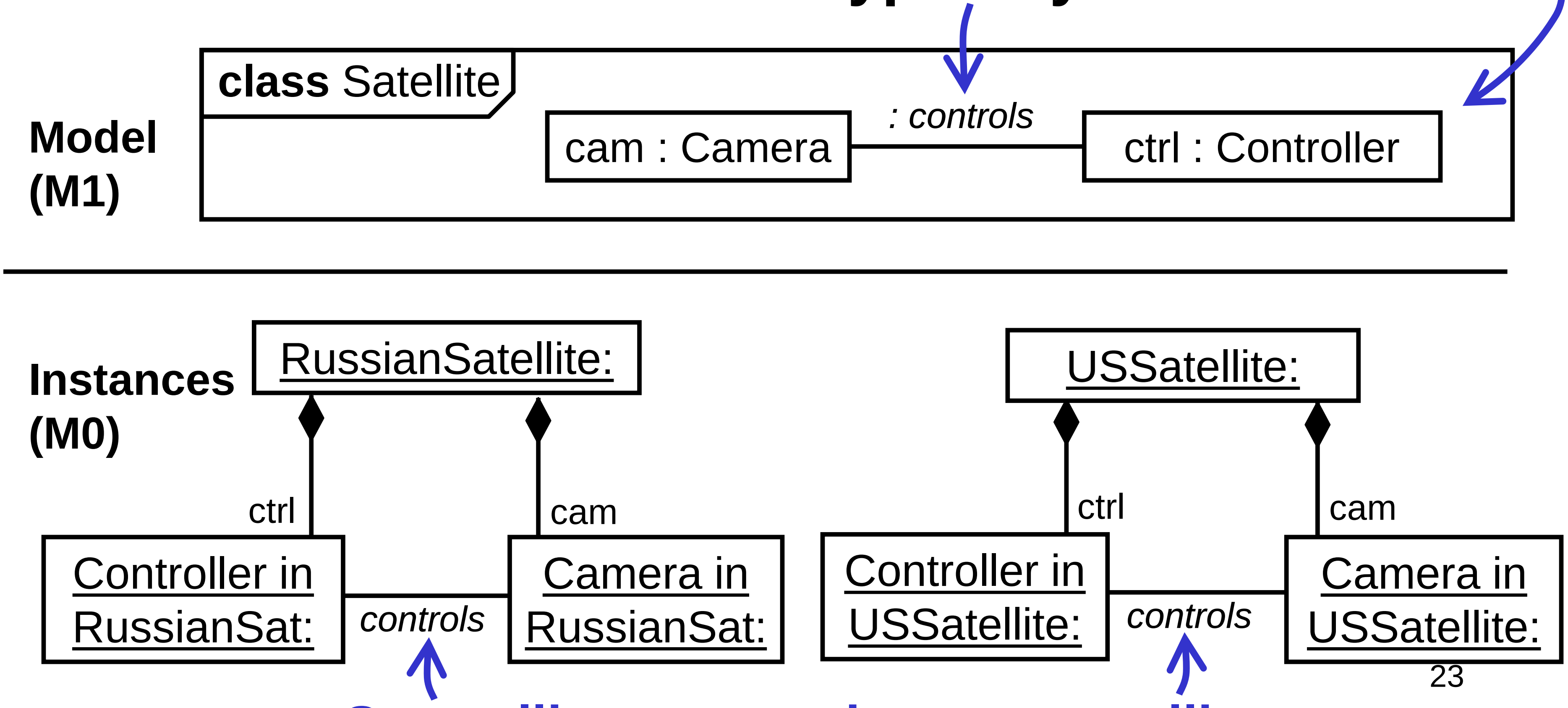
Composite Structure

- Whole-part relationships can modeled as associations, but part-part relationships cannot.



UML Composite Structure

- New diagram in UML 2.
  - Rectangles are properties typed by classes.
  - Lines are connectors typed by associations



Model (M1)

class Satellite

cam : Camera

: controls

ctrl : Controller

Instances (M0)

RussianSatellite:

ctrl

cam

Controller in RussianSat:

controls

Camera in RussianSat:

USSatellite:

ctrl

cam

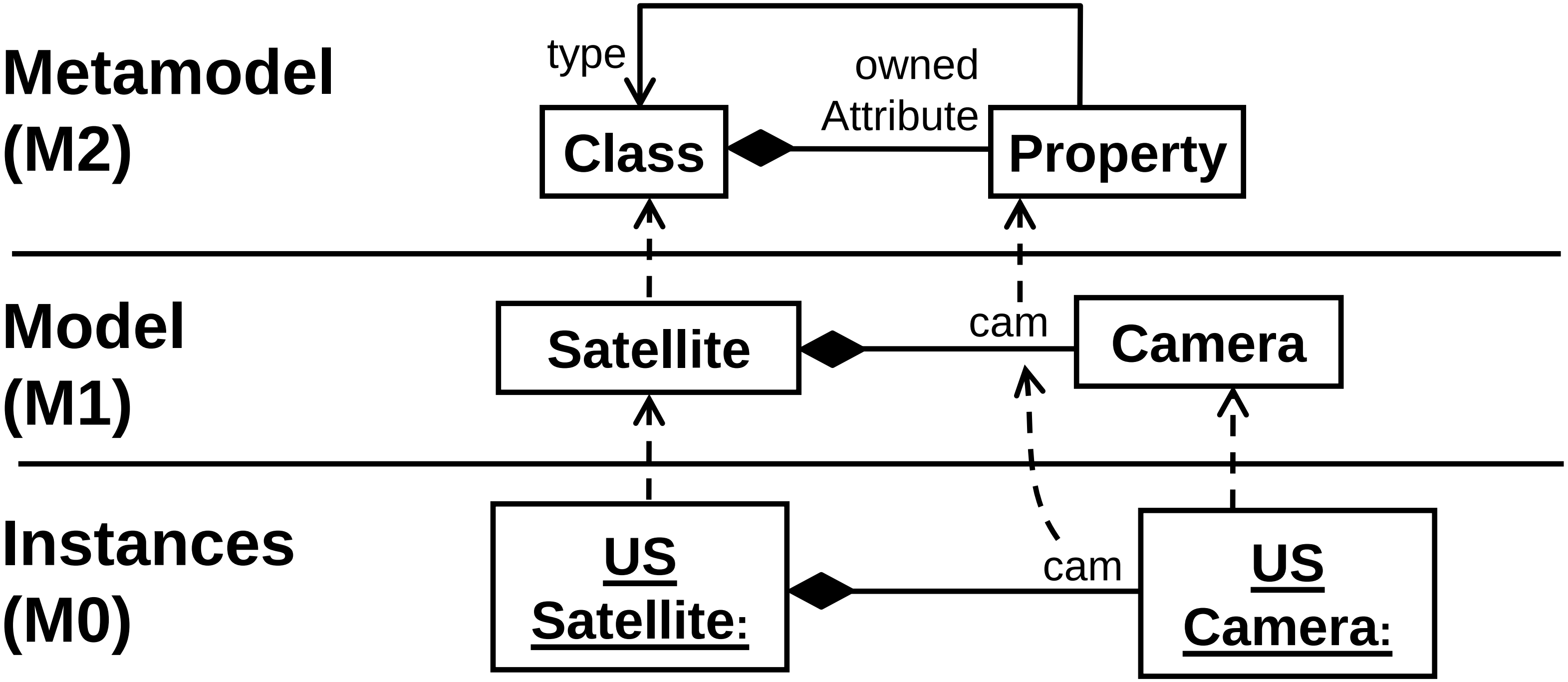
Controller in USSatellite:

controls

Camera in USSatellite:

Controlling camera in same satellite

Whole-part Metamodeling



Metamodel (M2)

type

owned Attribute

Class

Property

Model (M1)

Satellite

cam

Camera

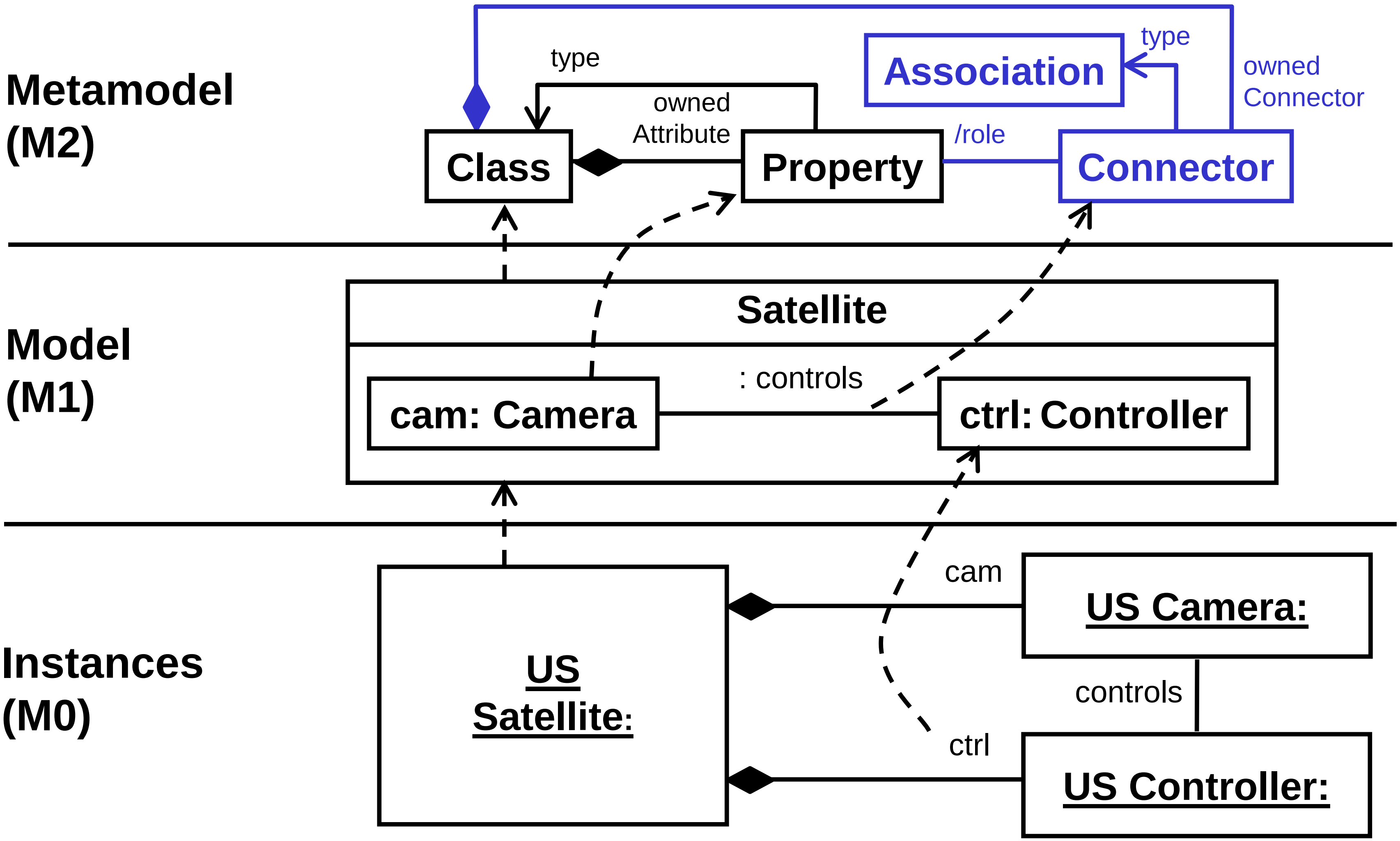
Instances (M0)

US Satellite:

cam

US Camera:

Part-part Metamodeling



Formal Languages for Composite Structure

- Area of ongoing research.
- Less restrictive specializations of first order logic (larger fragments).
- See complex role inclusion in OWL / SROIQ DL.
- Rule (non-monotonic) languages.

Behaviors as Composites

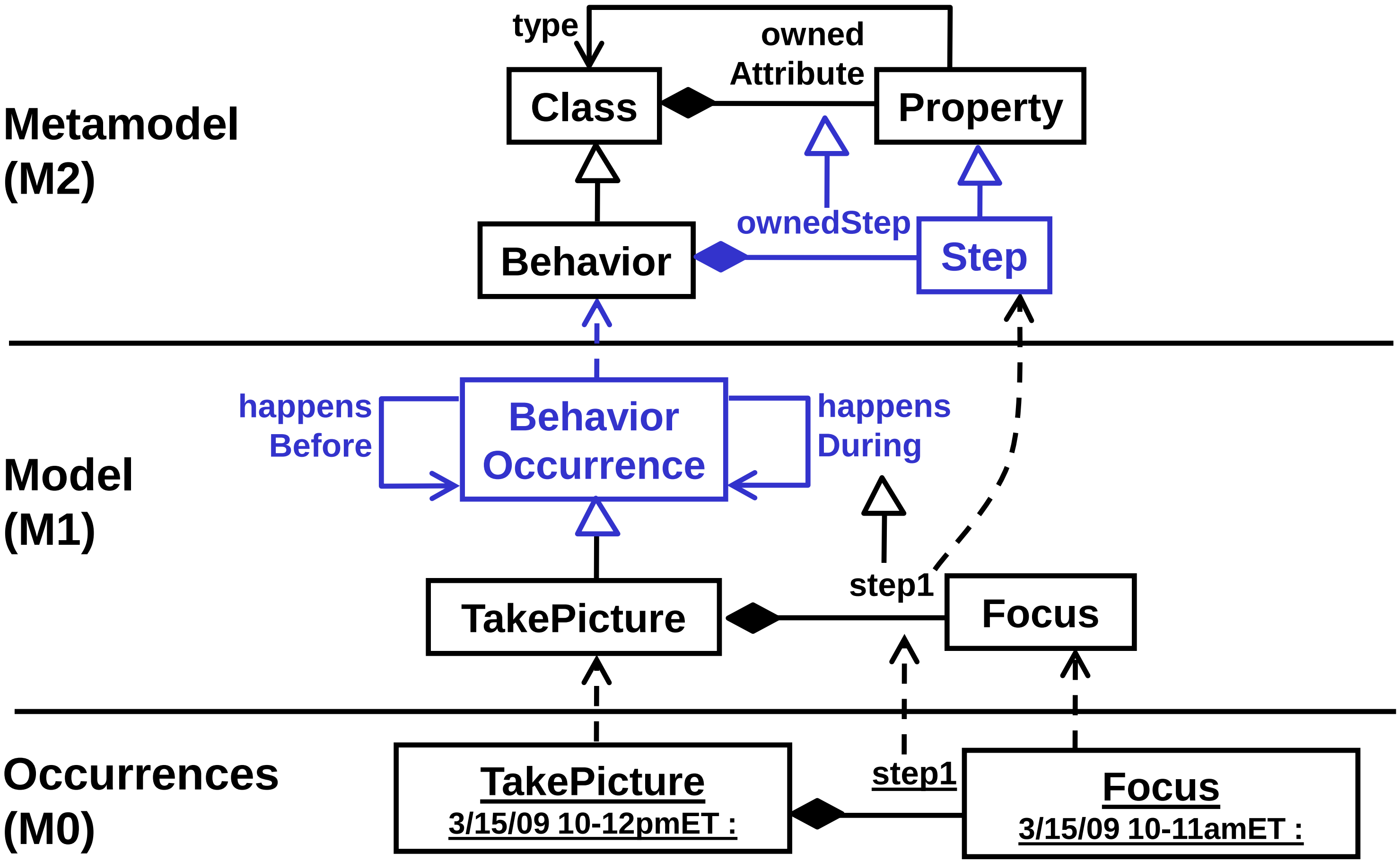
- Motivation
- Logical modeling
- Composite structure
- **Behaviors as composites**
  - **Sequencing**
    - Aside on SysML extensions for logical modeling
  - Events
  - Participants
  - Flows (object flows and messaging)
    - External participants
    - Flow ordering
    - Composition with flows and participants

UML Composite Structure Applied to UML Behaviors*

- **Whole-part**
  - Activities have actions, some calling behaviors.
  - State machines have state behaviors and submachines.
  - Interactions have interaction uses, messages, and actions.
- **Part-part**
  - Activities have control and object flow between actions.
  - State Machines have transitions between states.
  - Interactions have general orderings between messages.

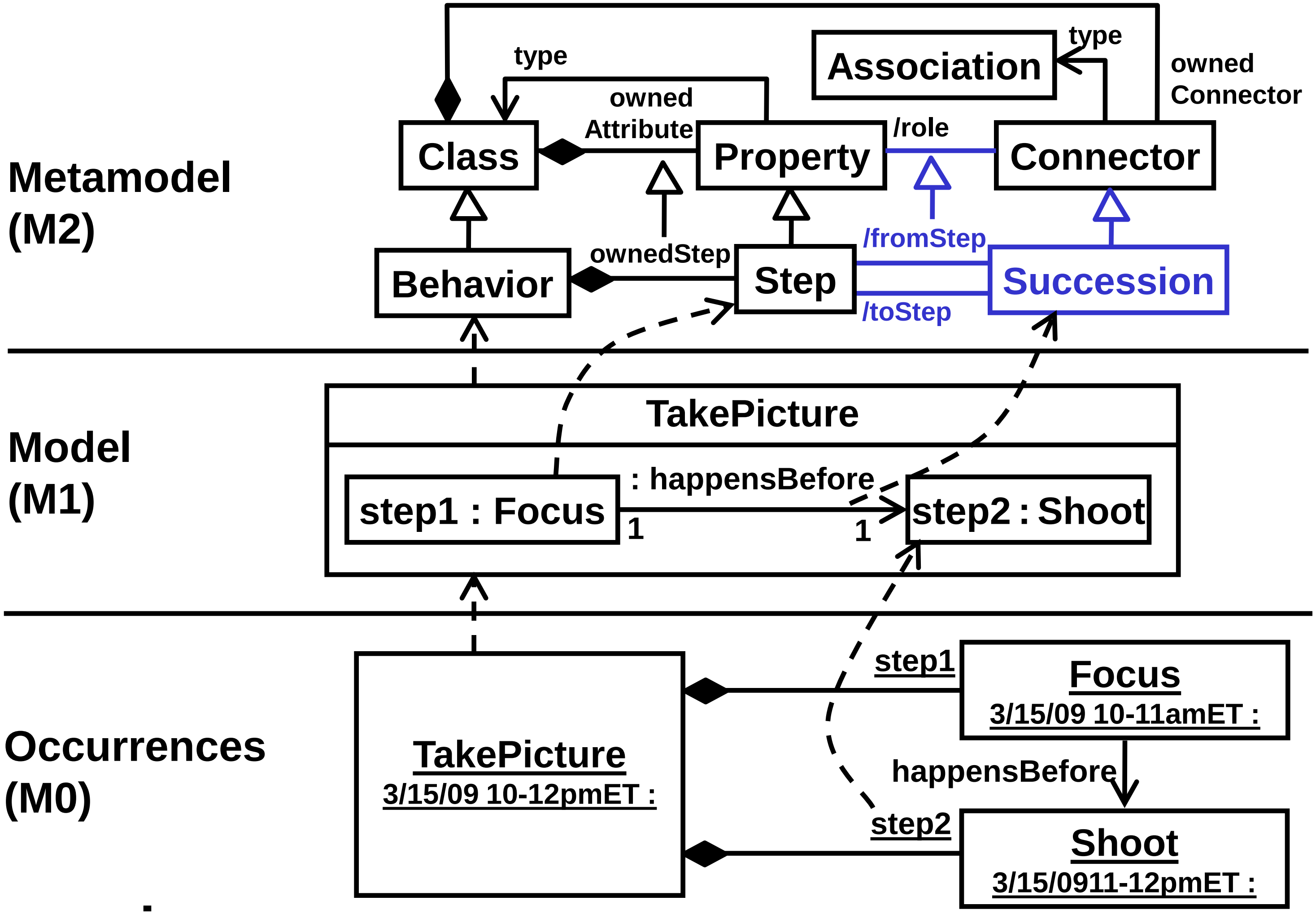
*** Not in UML, a bit in SysML.**

Whole-part for Behaviors



- **Steps:**
  - Are properties ...
  - typed by behaviors at M1, and specialized from a general temporal relation (happensDuring) …
  - that have “suboccurrences” as values at M0.

Part-part for Behaviors



- **Successions:**
  - Are connectors …
  - typed by general temporal relation at M1 (happensBefore) …
  - resulting in links between suboccurrences at M0.

Aside on SysML Extensions for Logical Modeling

- Motivation
- Logical modeling
- Composite structure
- Behaviors as composites
  - Sequencing
    - **Aside on SysML extensions for logical modeling**
  - Events
  - Participants
  - Flows (object flows and messaging)
    - External participants
    - Flow ordering
    - Composition with flows and participants

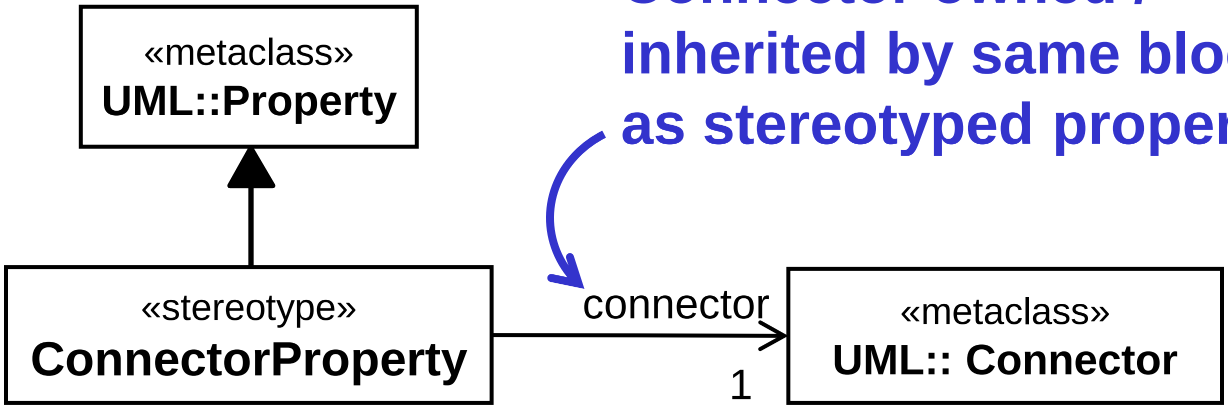
What’s Missing in UML?

- Properties for behavioral elements.
  - Connectors only link properties.
- UML does not have these.
  - Not even complete for logical structure modeling.
- Requires significant overhaul of UML metamodel.
  - To make some existing elements into properties and define their values.
- SysML working around this in some areas.

SysML’s Workaround Approach

- Extend UML properties.
- An example from logical structure modeling: Connector properties.

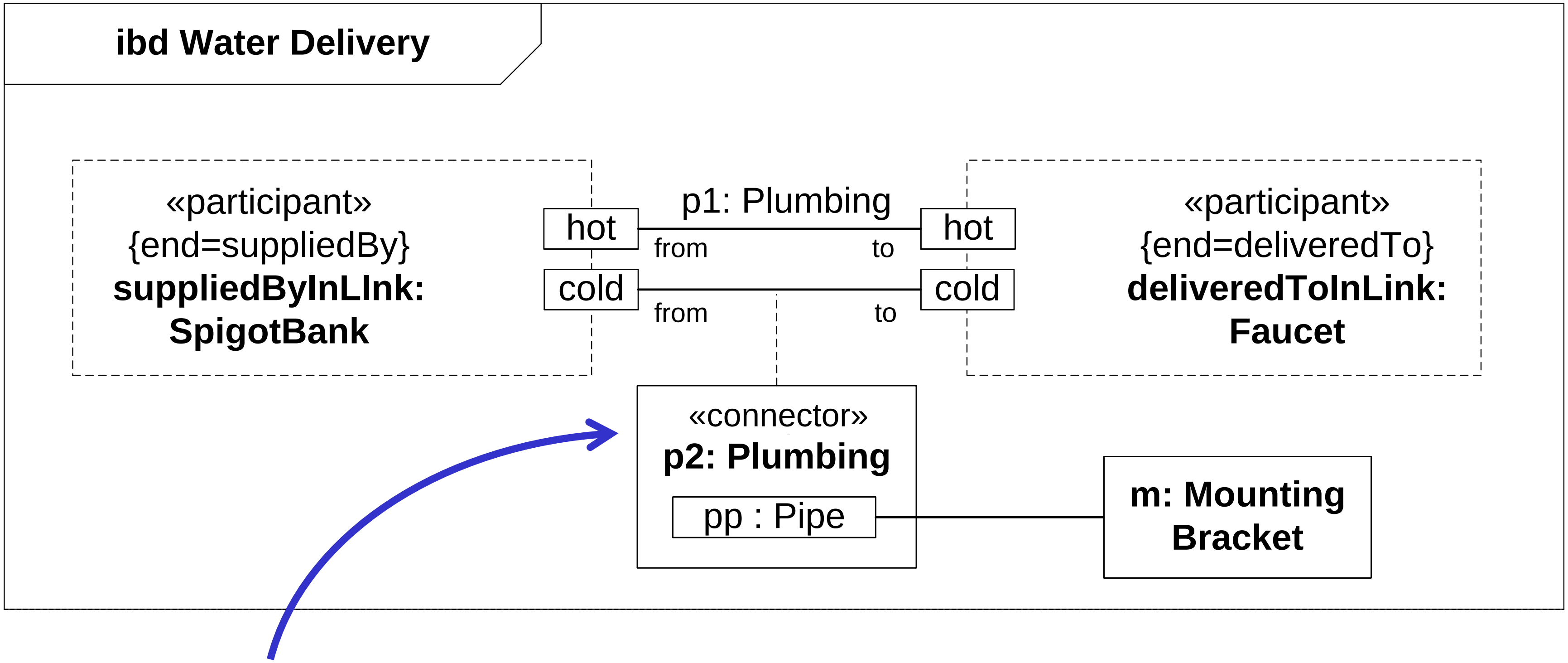
Property values = instances of association block typing connector that are created because of the connector.



Connector owned / inherited by same block as stereotyped property.

- Workaround: Keep properties and connectors in sync (“double-bookkeeping”).

Connector Properties

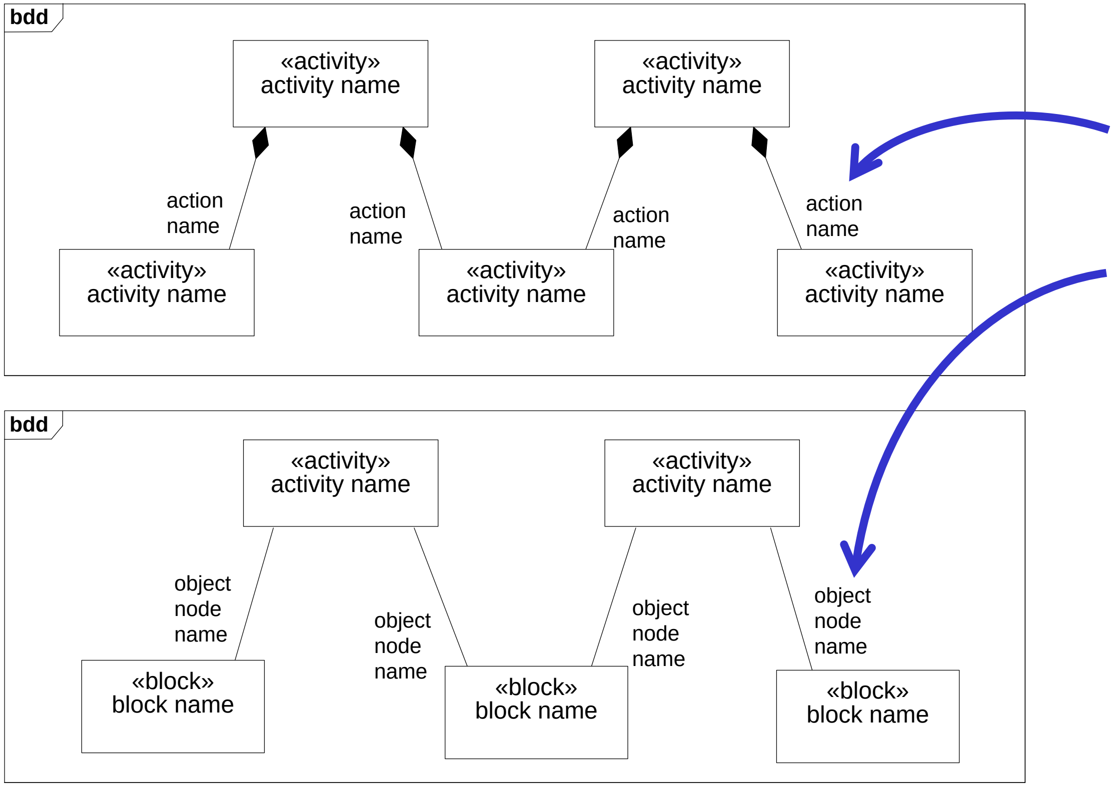


Connector properties, referring to connectors typed by association blocks.

Property values = instances of association blocks that are created because of the connector.

Logical Behavior Modeling in SysML

- Weakly addressed for call behavior actions and object nodes.
  - Diagram extensions for activities in BDDs.



Properties of activities with same names as actions and object nodes in that activity

Property values = instances of activities (executions) that are started due to calls and instances of object node types that are in the object node.

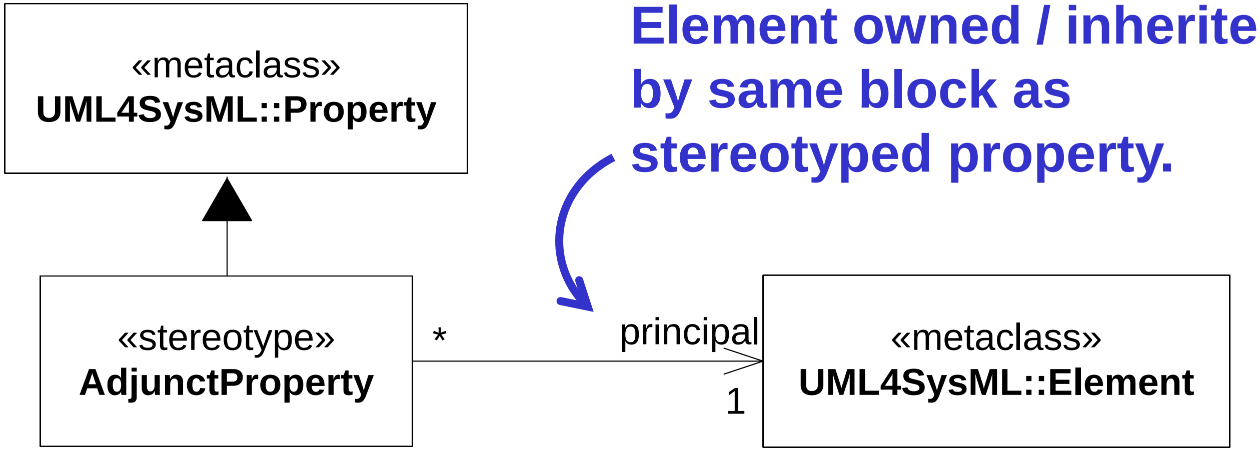
Logical Behavior Modeling not in SysML 1.3 or earlier

- Other elements that should be navigable as properties:
  - Parameters
  - Activity Variables
  - Submachine States
  - Interaction Uses
  - Many others

SysML 1.4: Adjunct Properties

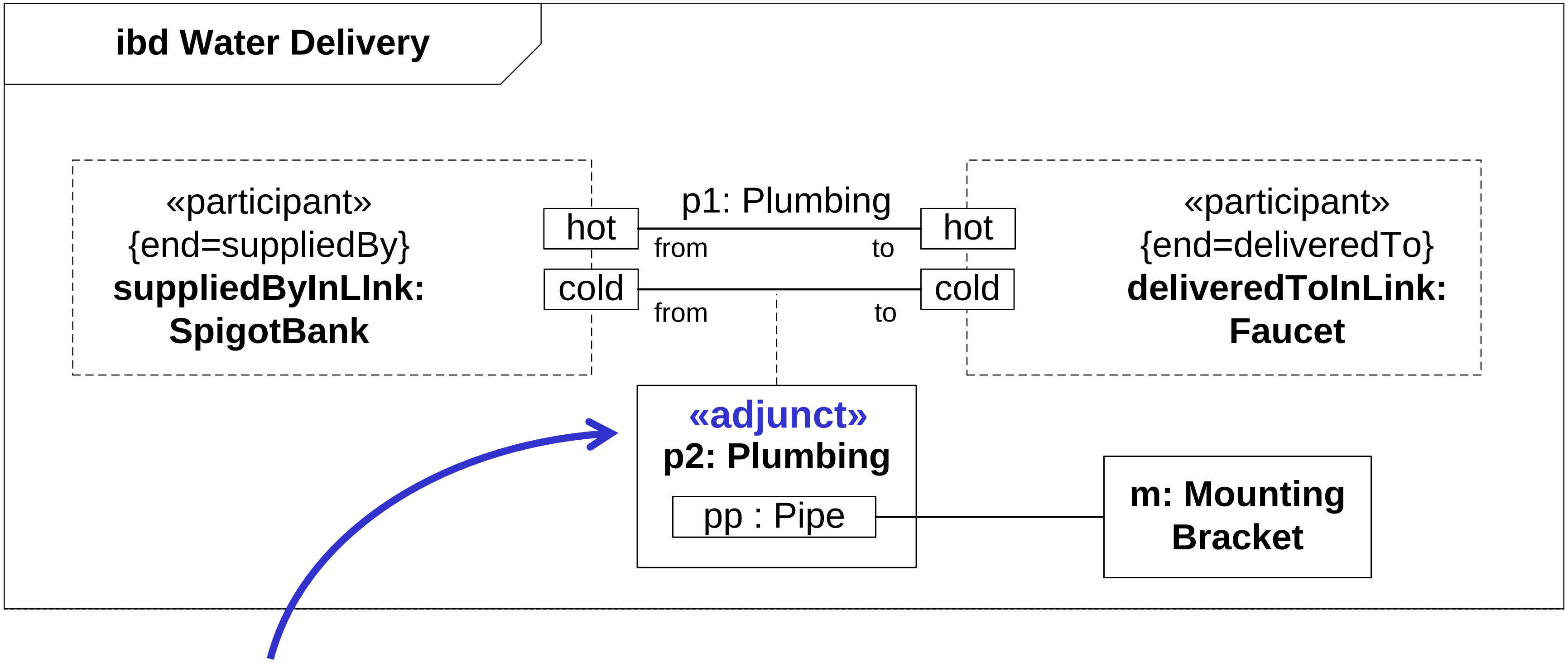
- Single stereotype applying to properties.
- Giving values for Call Actions, Object Nodes, Connectors, Parameters, Variables, Submachine States, Interaction Uses.

Property values = instances of element's type (type of parameter, object node, connector, behavior called, etc).



Element owned / inherited by same block as stereotyped property.

Adjunct Properties, Connectors

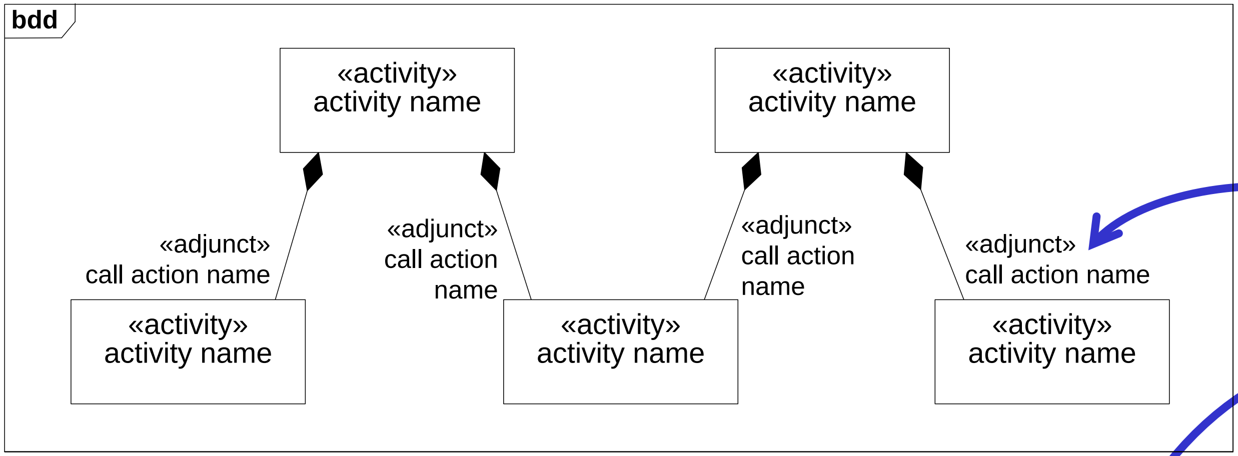


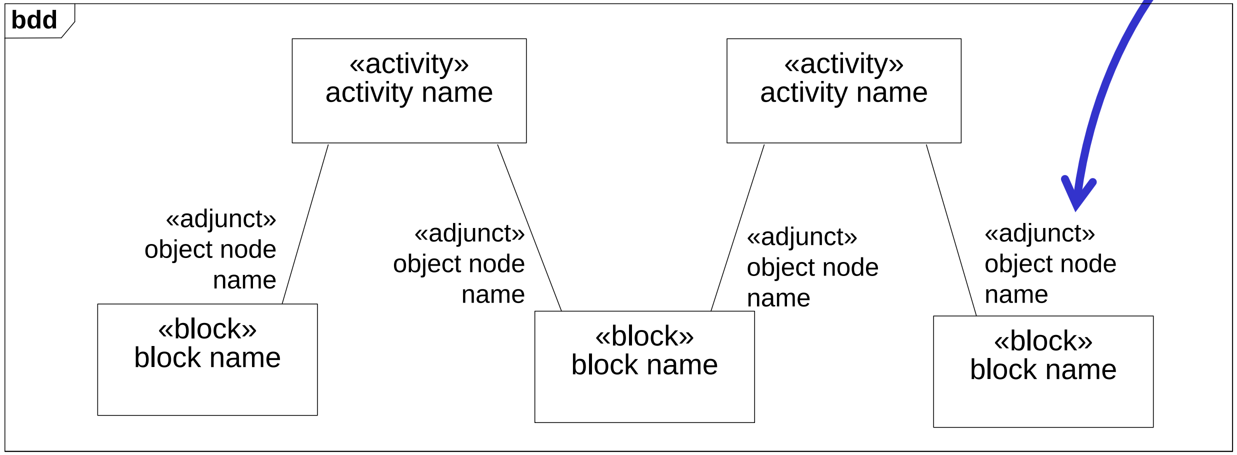
Adjunct properties, referring to connectors typed by association blocks.

Property values = instances of association blocks that are created because of the connector.

- Same as ConnectorProperty.
  - ConnectorProperty still in SysML 1.4.

APs, CallActions & ObjectNodes



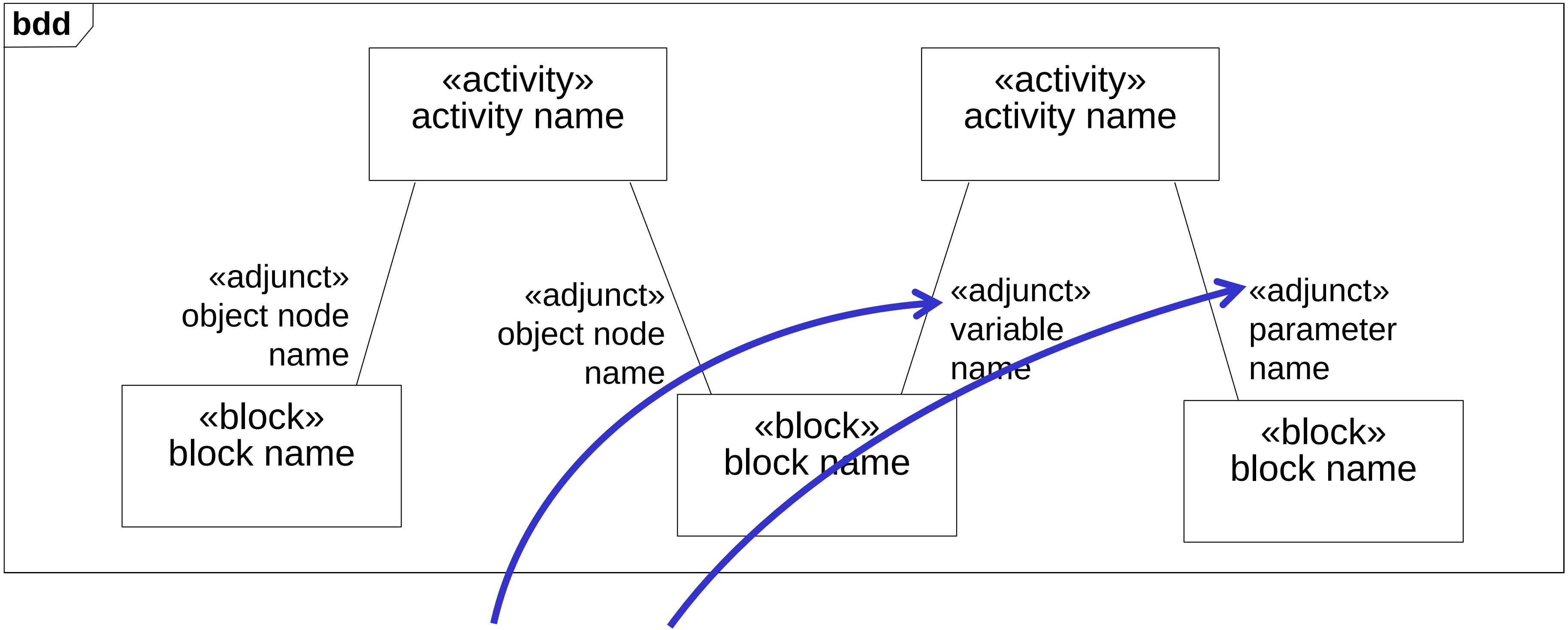


Adjunct properties of activities, referring to call behavior actions and object nodes.

Property values = instances of activities (executions) that are started due to calls and instances of object node types that are in the object node.

- **Not dependent on name matching.**

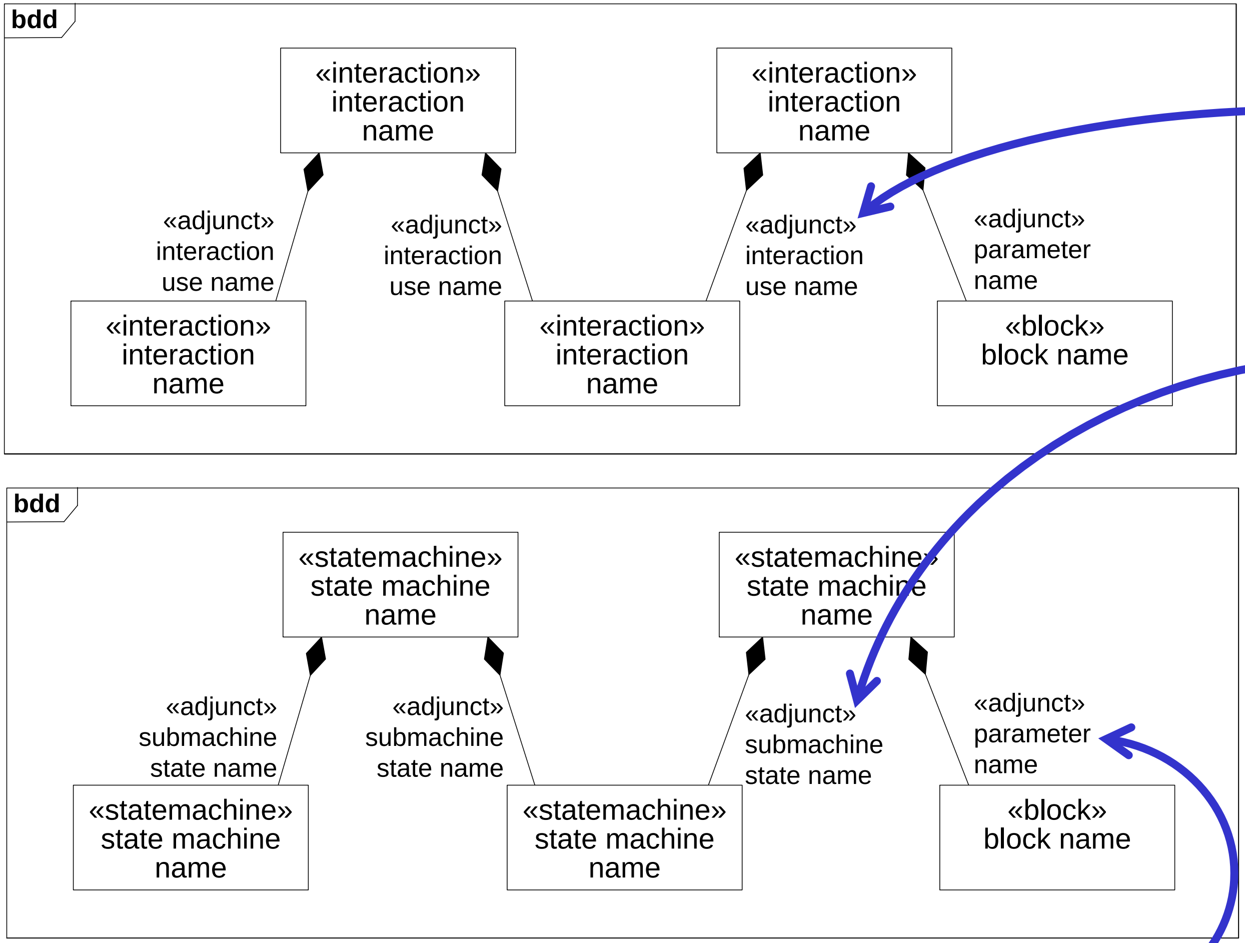
APs, Parameters & Variables



Adjunct properties of activities, referring to variables and parameters.

Property values = instances of variable or parameter type that are assigned to the variables or parameters.

APs,“Calls” & Parameters on SM and Int

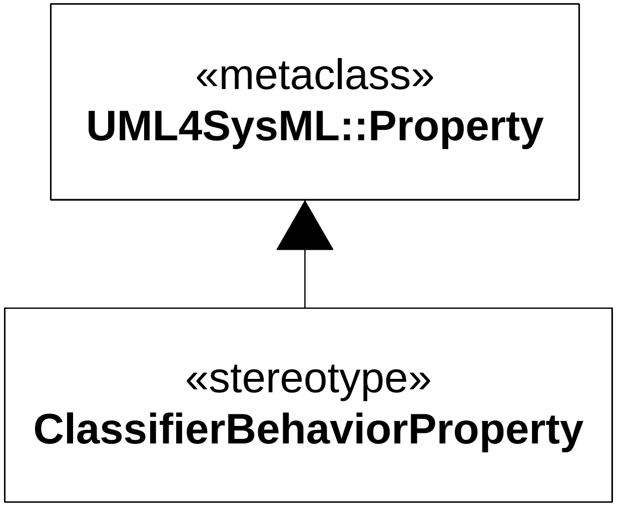


Adjunct properties of interactions and state machines, referring to submachine states and interaction uses.

Property values = instances of state machines or interactions (executions) that are started due to interaction uses and submachine states.

- Similarly for parameters.
  - Property values = instances of parameter type that are assigned to the parameters.

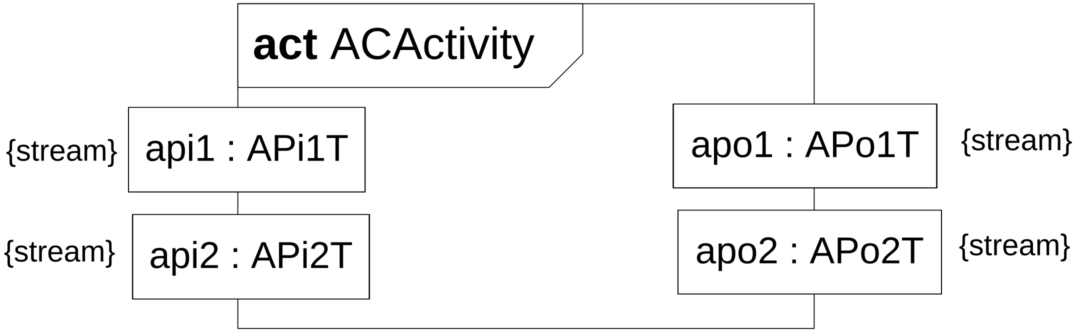
SysML 1.4: Classifier Behavior Properties

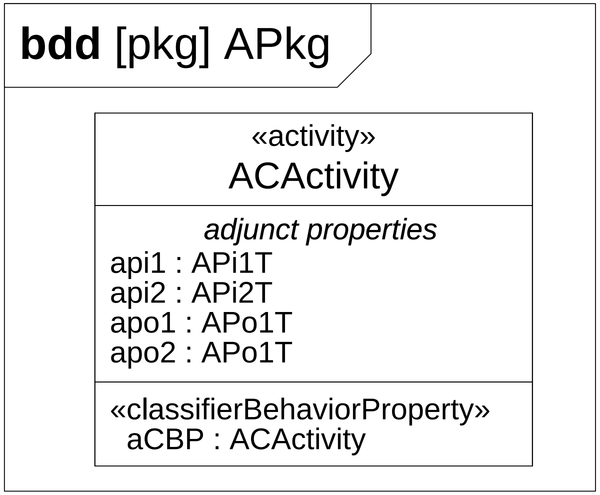


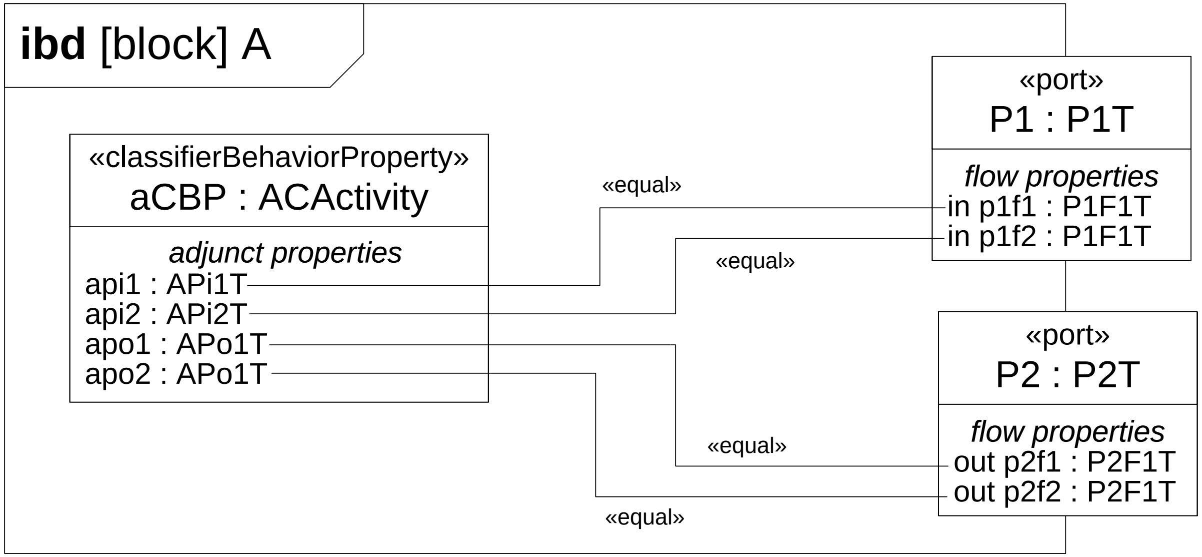
- **Value is the execution of the classifier behavior in an instance of a block.**
  - **Enables connectors to properties of classifier behaviors, such as adjuncts for parameters**

Example Applying CBPs and APs

- **Binding parameters to flow properties on block or ports.**







Returning to Behaviors as Composites

- Motivation
- Logical modeling
- Composite structure
- **Behaviors as composites**
  - Sequencing
    - Aside
  - **Events**
  - **Participants**
  - **Flows (object flows and messaging)**
    - **External participants**
    - **Flow ordering**
    - **Composition with flows and participants**

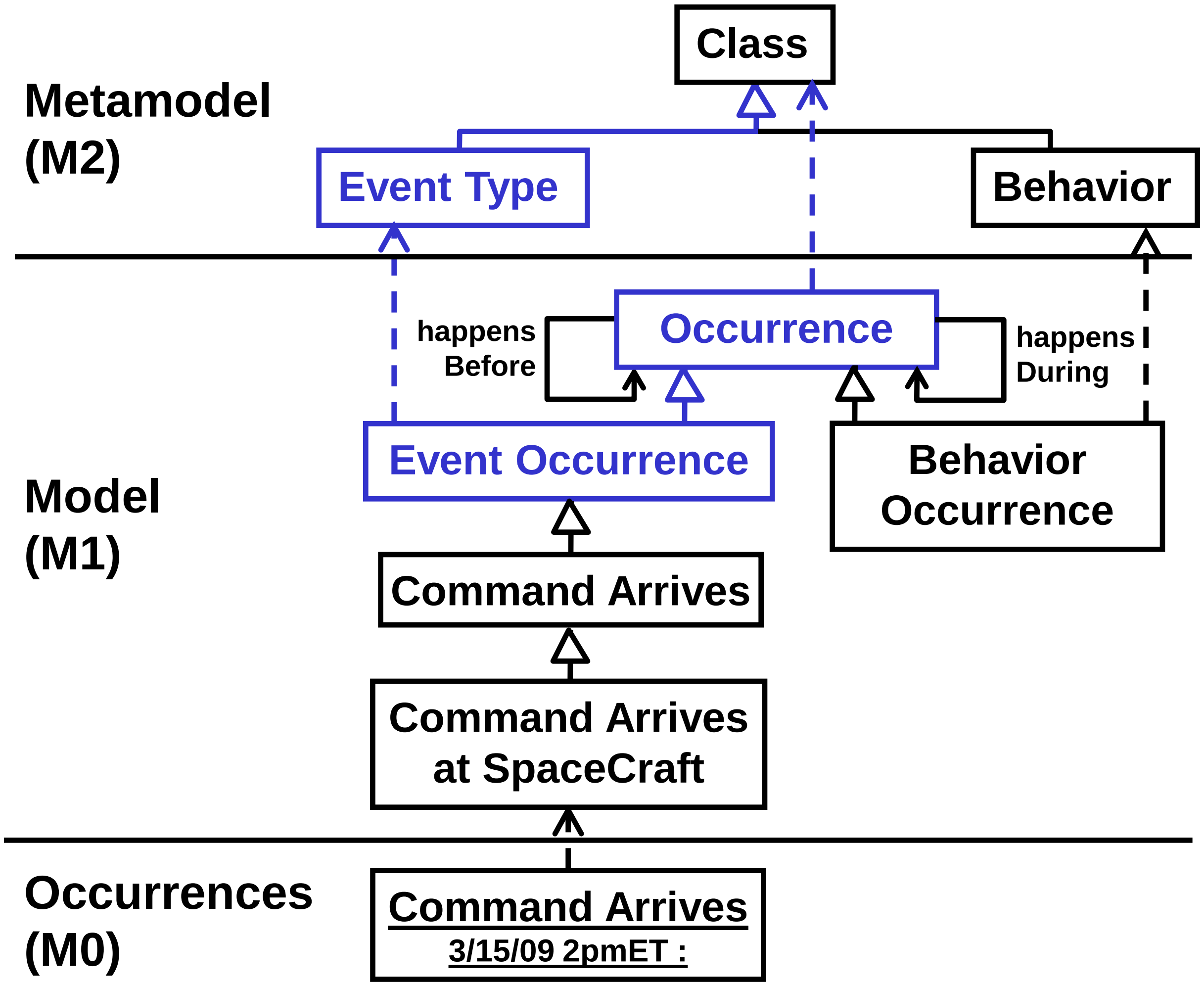
Events

- Motivation
- Logical modeling
- Composite structure
- Behaviors as composites
  - Sequencing
    - Aside
  - **Events**
  - Participants
  - Flows (object flows and messaging)
    - External participants
    - Flow ordering
    - Composition with flows and participants

Events

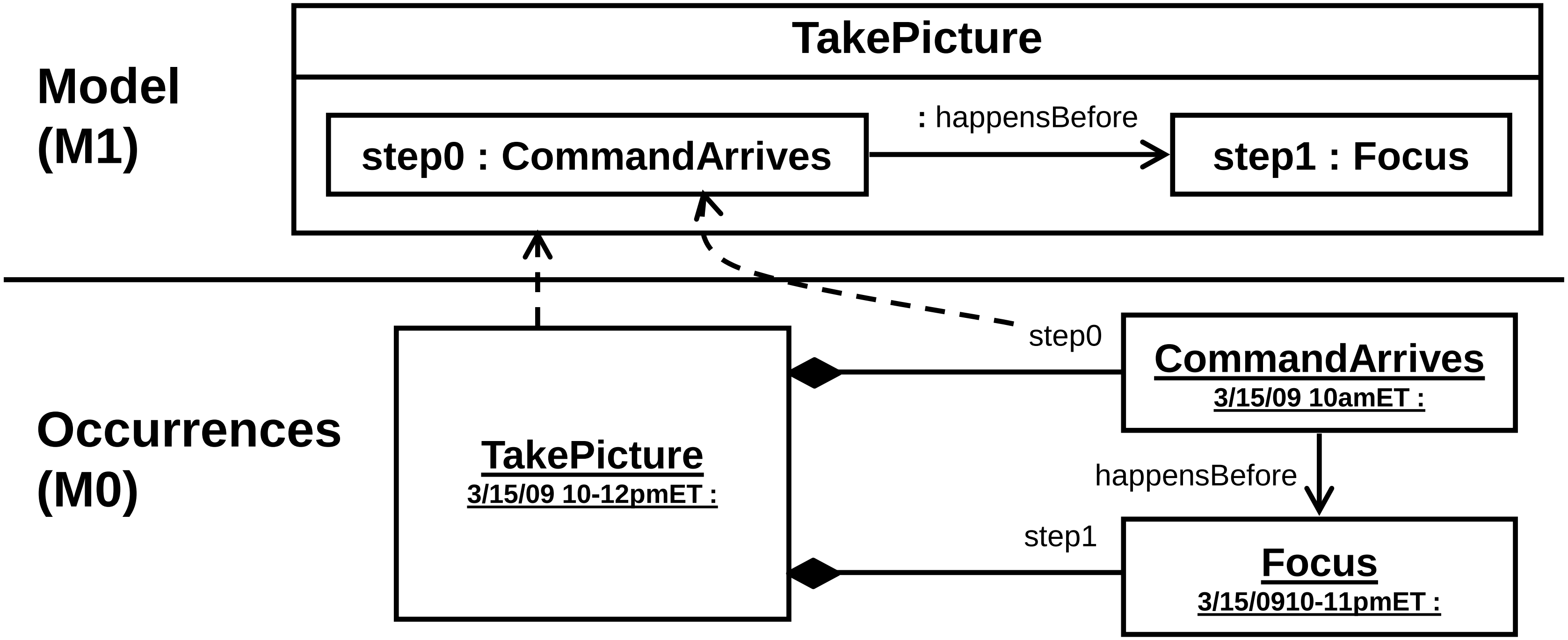
- Events are alot like behaviors.
  - They occur at particular times at M0.
  - Can be specified by types at M1, which can be subtyped.
  - Can be parts of behaviors.
  - Can be specified to happen in a certain order under those behaviors.

Event Types and Occurrences



- **Event types:**
  - **Are classes …**
  - **specialized at M1 (temporal relations promoted) …**
  - **with instances occurring at M0.**

Events in Behaviors



Model (M1)

TakePicture

step0 : CommandArrives

: happensBefore

step1 : Focus

Occurrences (M0)

TakePicture
3/15/09 10-12pmET :

step0

CommandArrives
3/15/09 10amET :

happensBefore

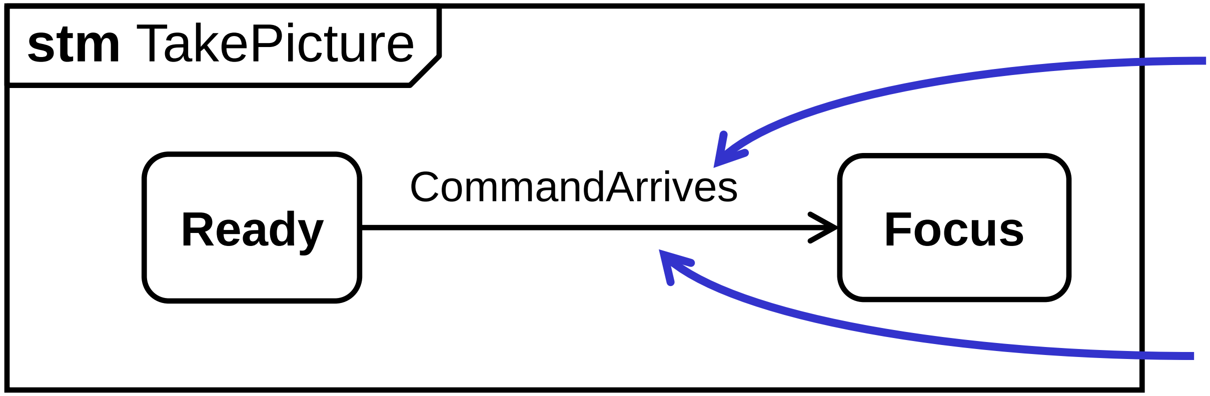
step1

Focus
3/15/0910-11pmET :

- Event types can be types of properties …
- … ordered by successions.

UML Event Notations

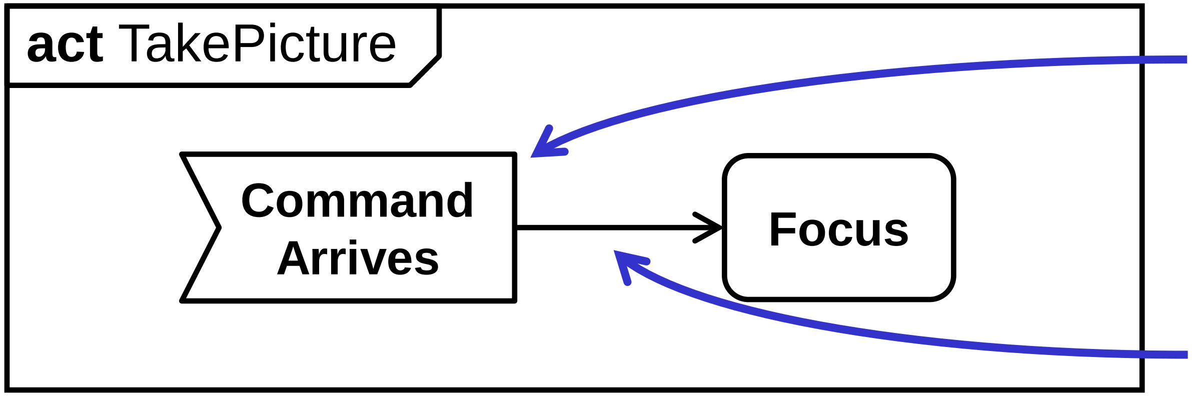
State Machine



Detects event occurrence

Points to step after event occurs (states have behavior)

Activity

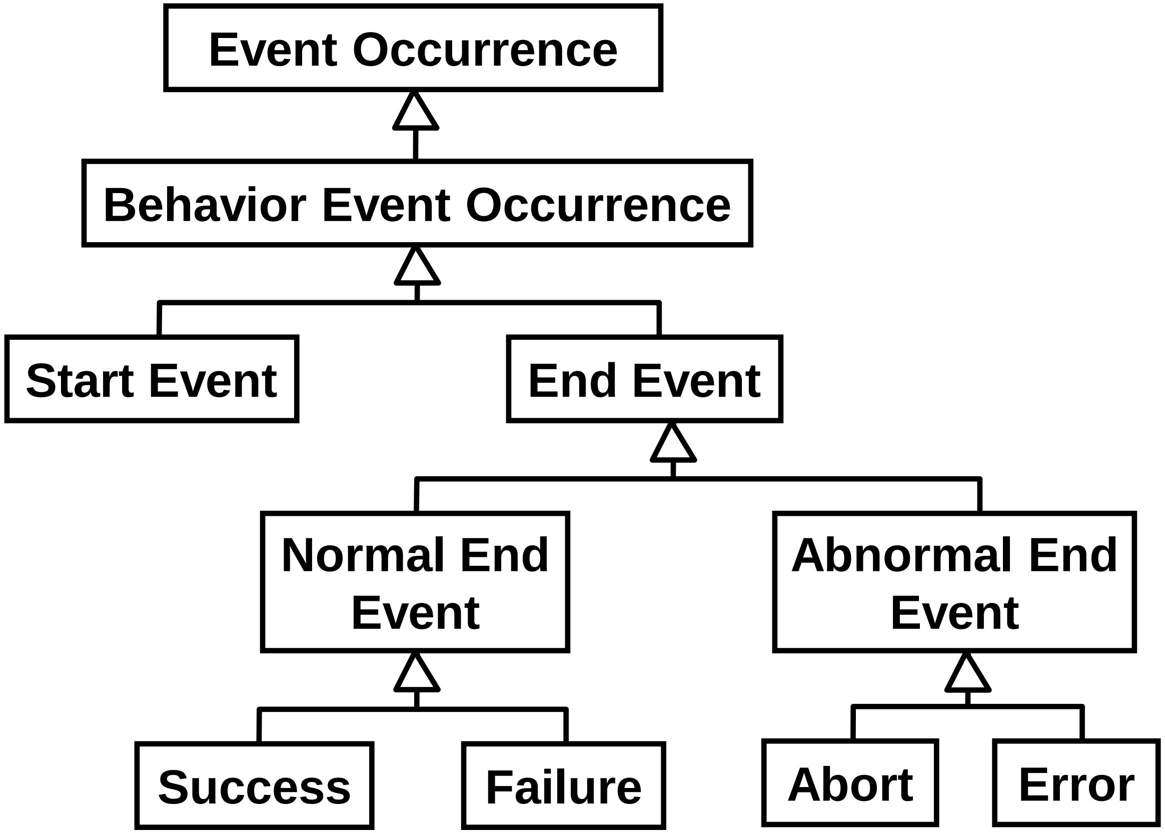


Detects event occurance

Points to step after event occurs

Behavior Events

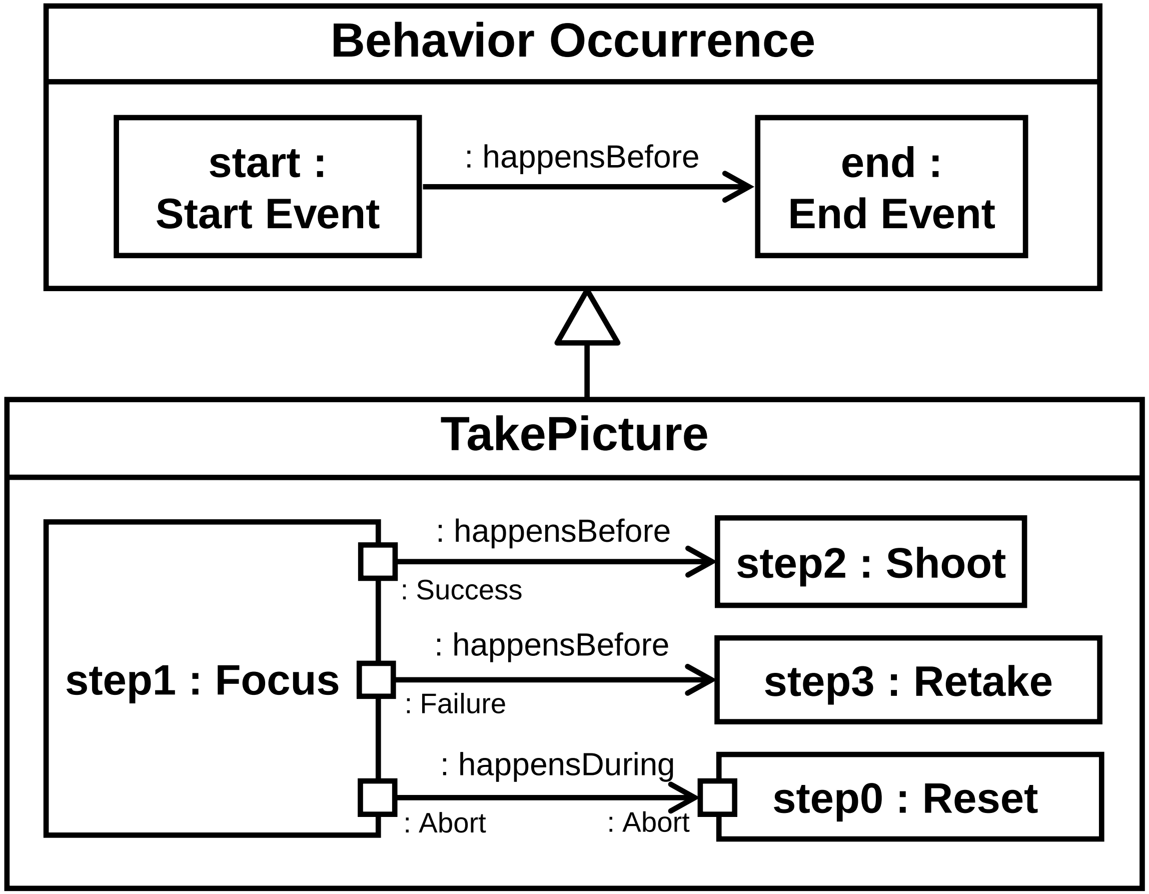
Model
(M1)



- Behaviors have specialized events for their lifecycle ...

Behavior Events as Parts

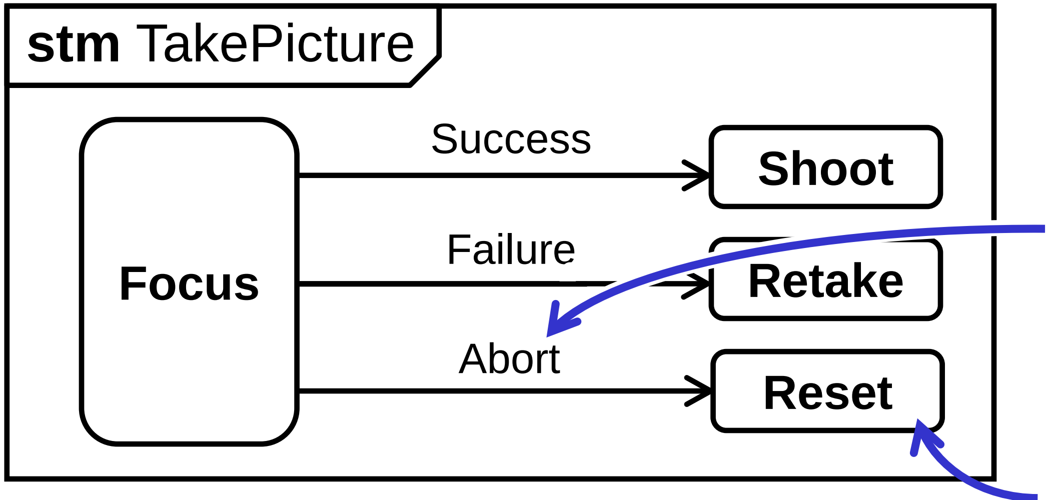
Model (M1)



- … which can by the types of “port” properties ...
- … that are ordered by successions.

UML Event Notations

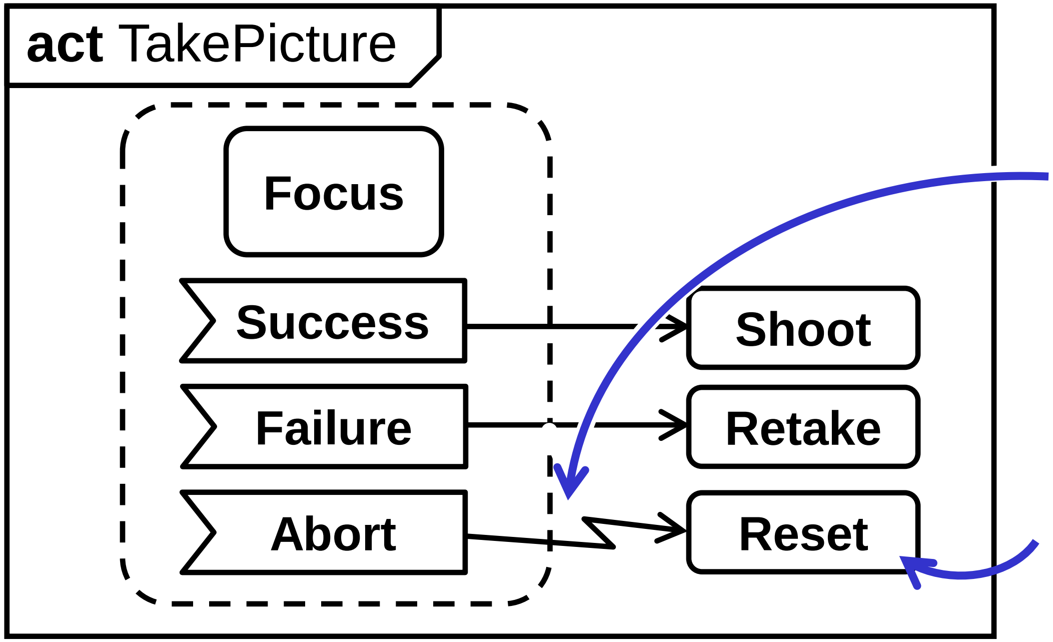
State Machine



Transitions abort source state behaviors (if the behaviors aren’t finished already).

Behaviors can’t happen in multiple states at once (Reset can’t cause abort).

Activity



Aborts behaviors and event detection occurring in region.

Behaviors can happen in multiple actions at once (Reset can cause abort, but will cause another reset).

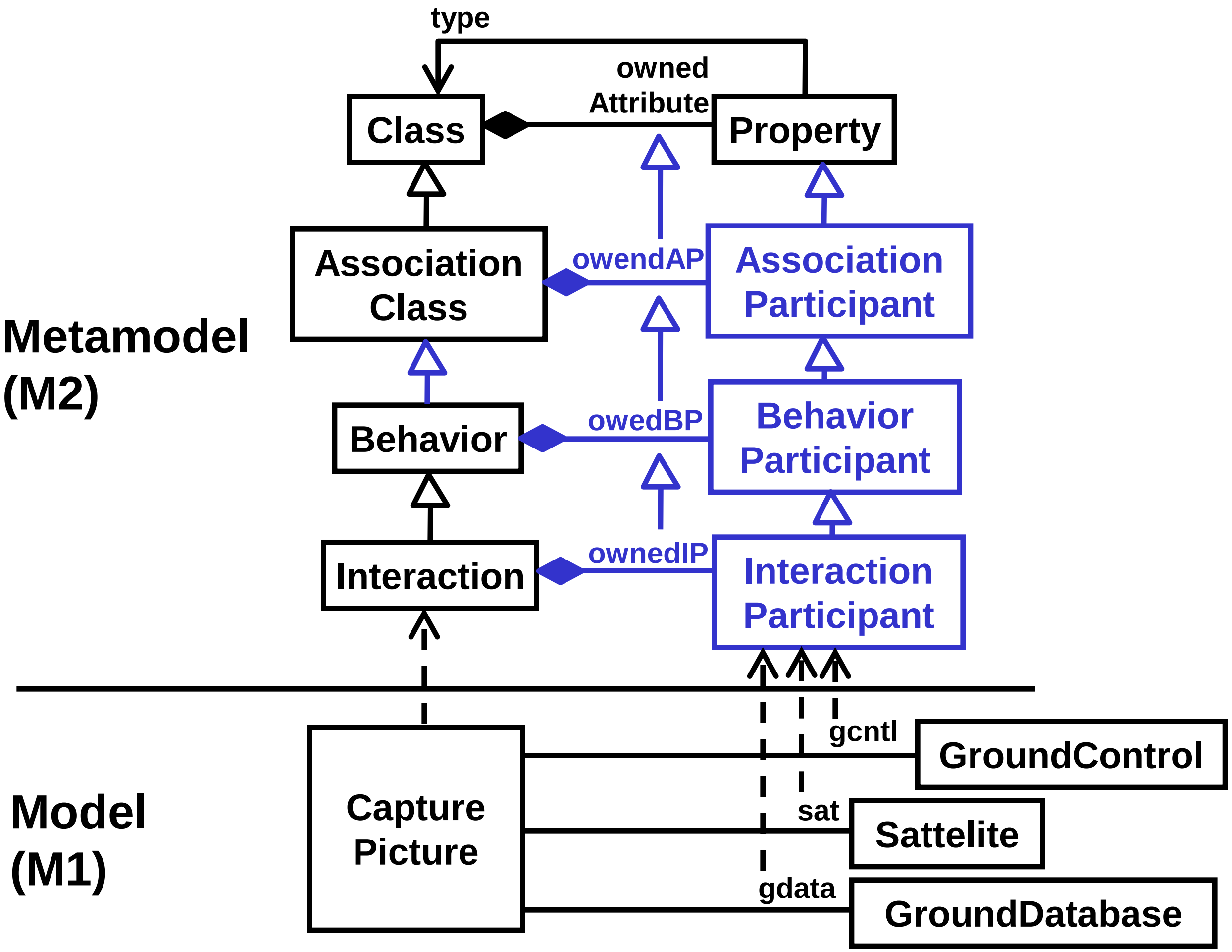
Participants

- Motivation
- Logical modeling
- Composite structure
- Behaviors as composites
  - Sequencing
    - Aside
  - Events
  - **Participants**
  - Flows (object flows and messaging)
    - External participants
    - Flow ordering
    - Composition with flows and participants

Participants

- Behaviors involve objects (that behave).
  - Interactions have lifelines.
  - Activities have object nodes, variables, and partitions.
  - Behaviors have parameters.
- Association involve objects that are linked.
- Behaviors are associations between their participants.

Participant Properties



- **Participants:**
  - **Are properties …**
  - **assigned participant types at M1 …**
  - **with individual values at M0 on occurrences / links.**

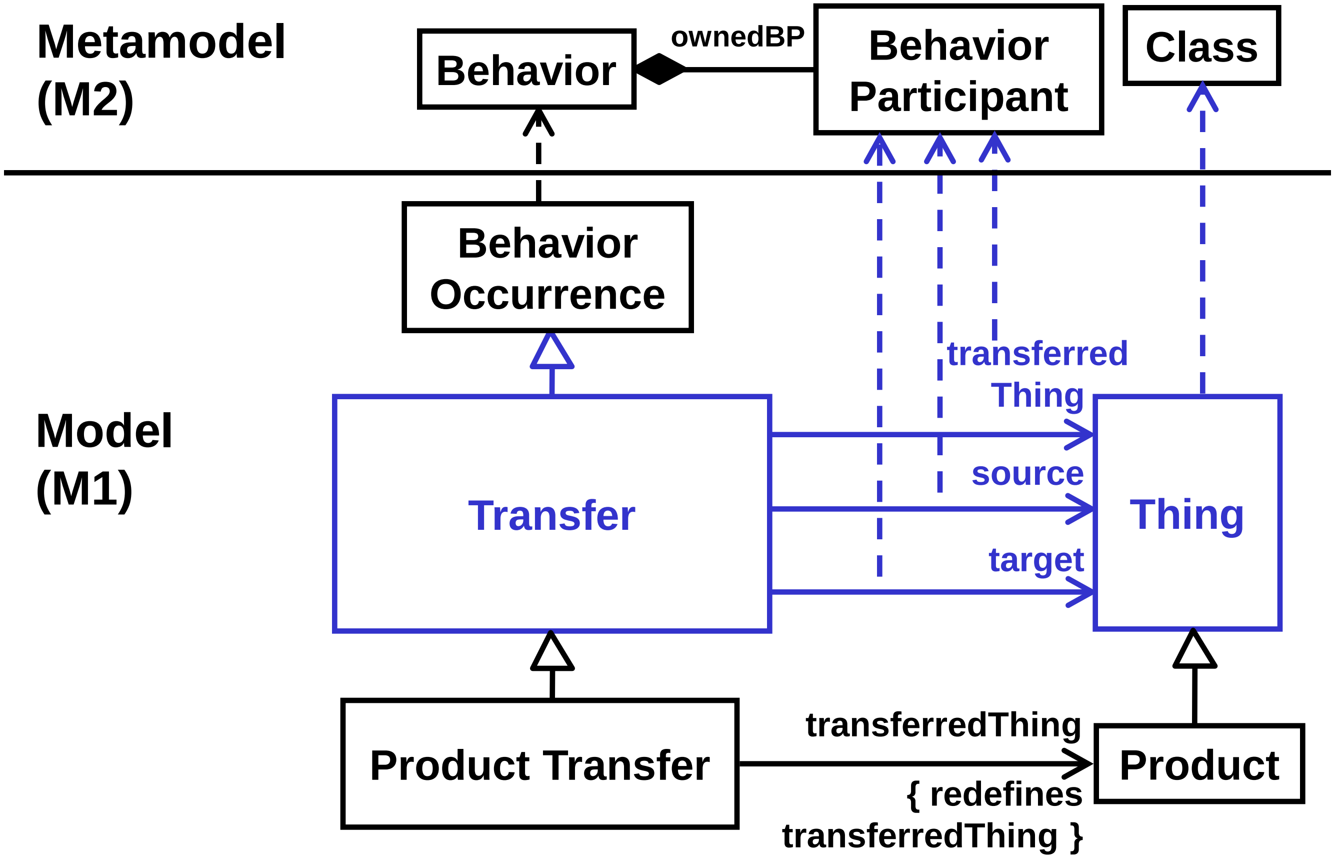
Flows (object flows and messaging)

- Motivation
- Logical modeling
- Composite structure
- Behaviors as composites
  - Sequencing
    - Aside
  - Events
  - Participants
  - **Flows (object flows and messaging)**
    - External participants
    - Flow ordering
    - Composition with flows and participants

Object Flows and Messaging

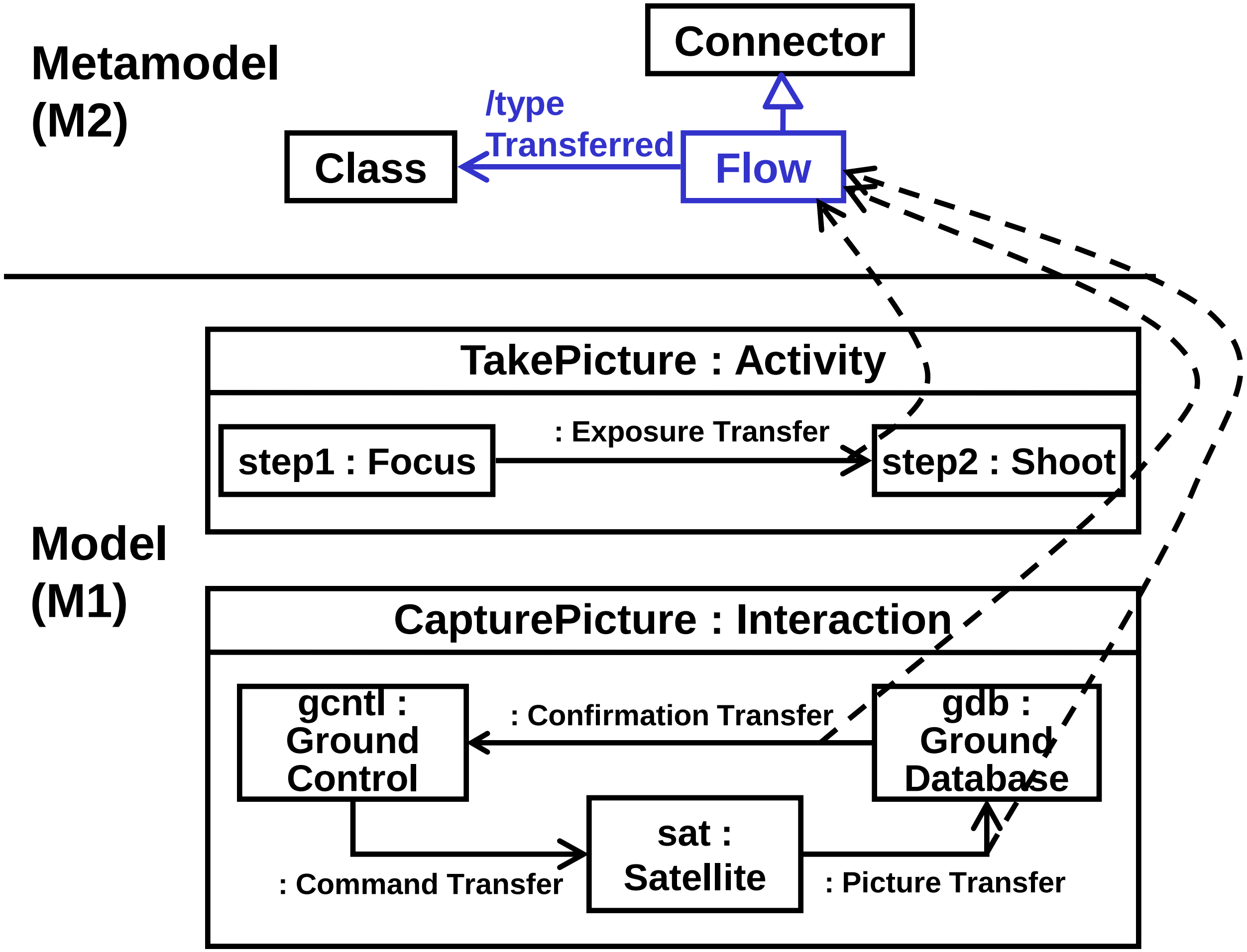
- Specify transfer of entities at M0.
  - Activities have object flows linking pins on actions.
  - Interactions have messages linking lifelines.
- Transfers take time, they are behavior occurrences.
  - Start when entity begins flowing, or message leaves the sender.
  - End when entity stops flowing, or message arrives at receiver.

Transfers



- **Transfers:**
  - **Are behavior occurrences ...**
  - **with participant properties, and are specialized at M1.**
  - **that occur at M0 involving individuals that are values of participant properties.**

Flows

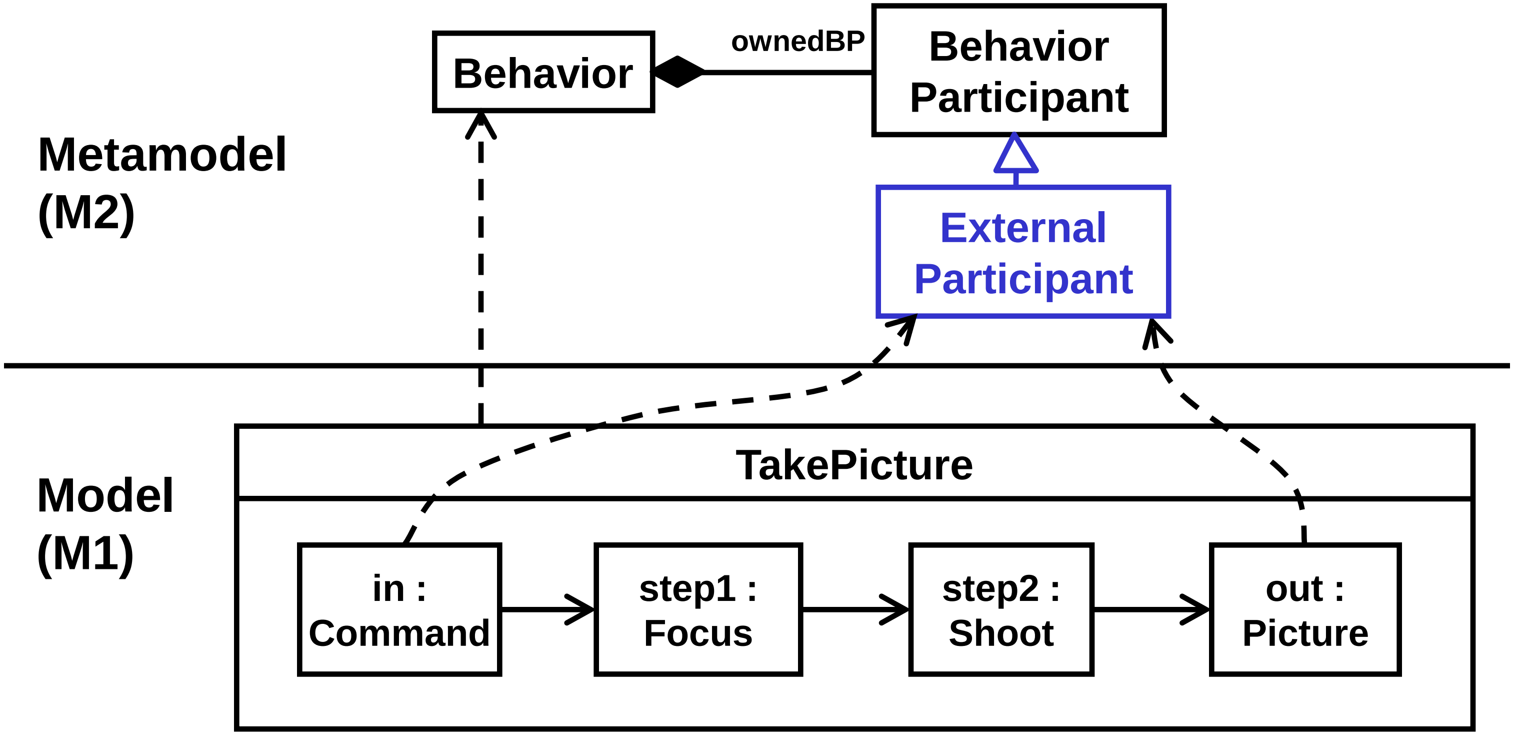


- Flows:
  - are connectors …
  - typed by transfers at M1 …
  - that have transfer occurrences as values at M0.

External Participants

- Motivation
- Logical modeling
- Composite structure
- Behaviors as composites
  - Sequencing
    - Aside
  - Events
  - Participants
  - Flows (object flows and messaging)
    - **External participants**
    - Flow ordering
    - Composition with flows and participants

External Participants



- External participants:
  - Are behavior participants …
  - that can be linked by flows at M1 for inputs and outputs ….
  - resulting in occurrences of transferring at M0.

Flow Ordering

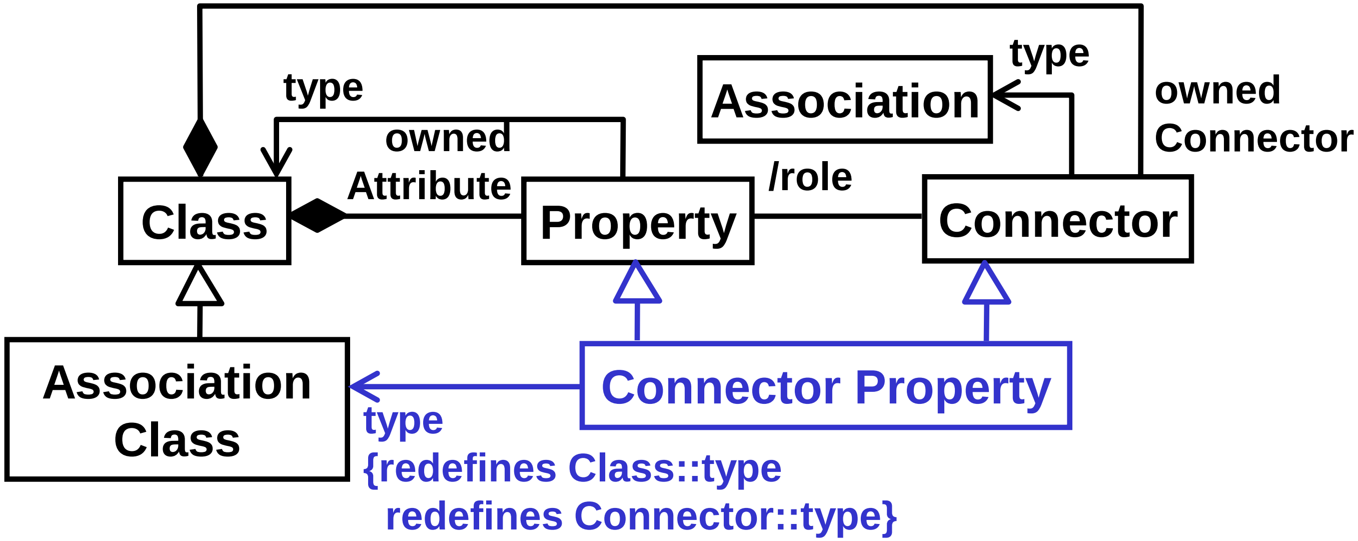
- Motivation
- Logical modeling
- Composite structure
- Behaviors as composites
  - Sequencing
    - Aside
  - Events
  - Participants
  - Flows (object flows and messaging)
    - External participants
    - **Flow ordering**
    - Composition with flows and participants

Flow Ordering

- Some flows happen before others
  - Interactions order messages and interaction uses.
  - Protocol state machines specify allowed orders of operation calls and other protocols.
  - Activities order actions for sending and receiving messages.
- Requires successions between flows (connectors between connectors).

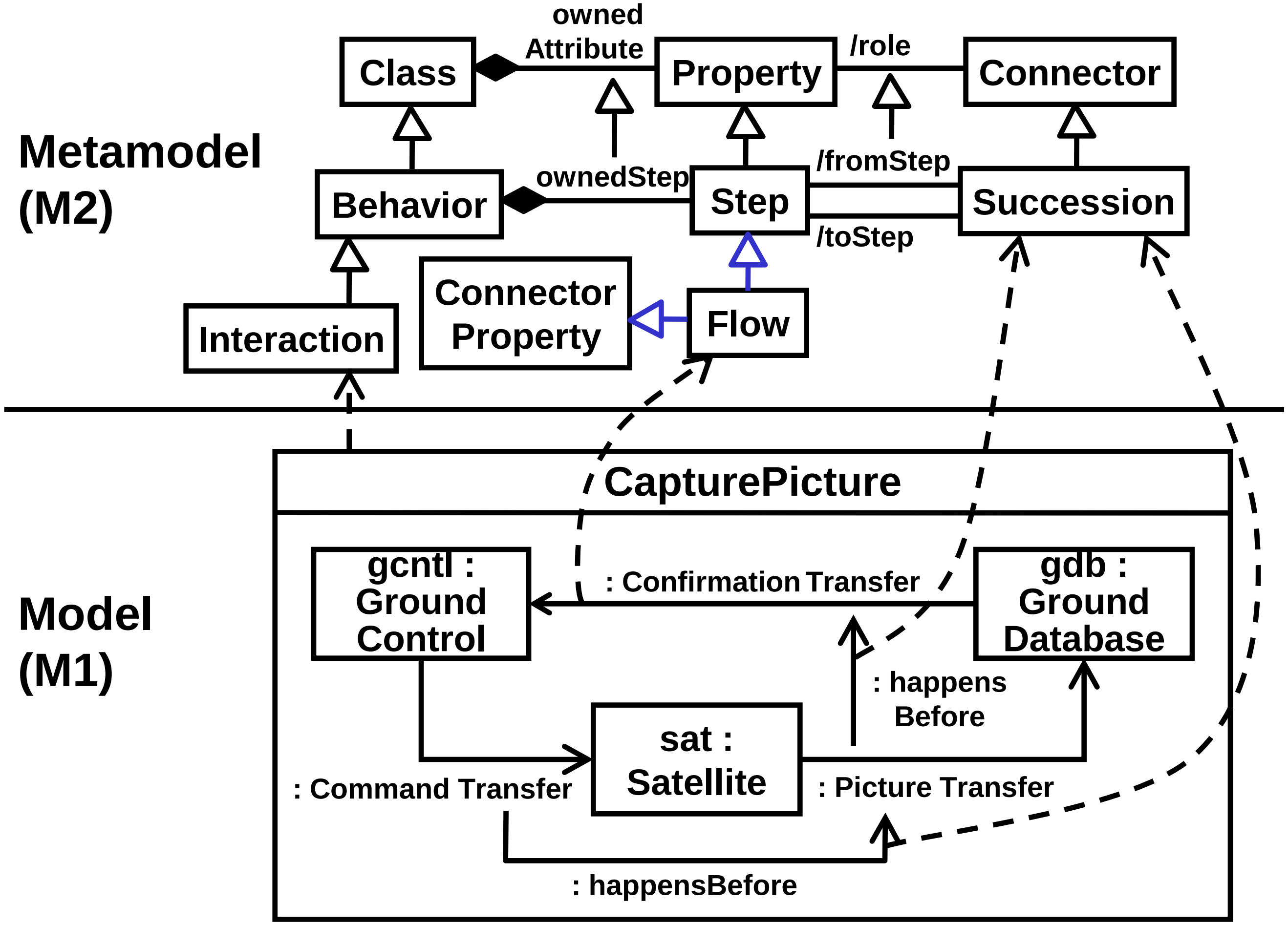
Connector Properties

Metamodel (M2)



- **Connector Properties:**
  - **Are connectors and properties at the same time ...**
  - **that have association classes as types at M1 …**
  - **and links as values at M0.**
- **M0 values of connector properties are links specified by connectors.**

Flow Properties



- **Flows:**
  - **Are connectors and steps at the same time …**
  - **connected by successions at M1 …**
  - **with transfer occurrences as values at M0.**

Composition with Flows and Participants

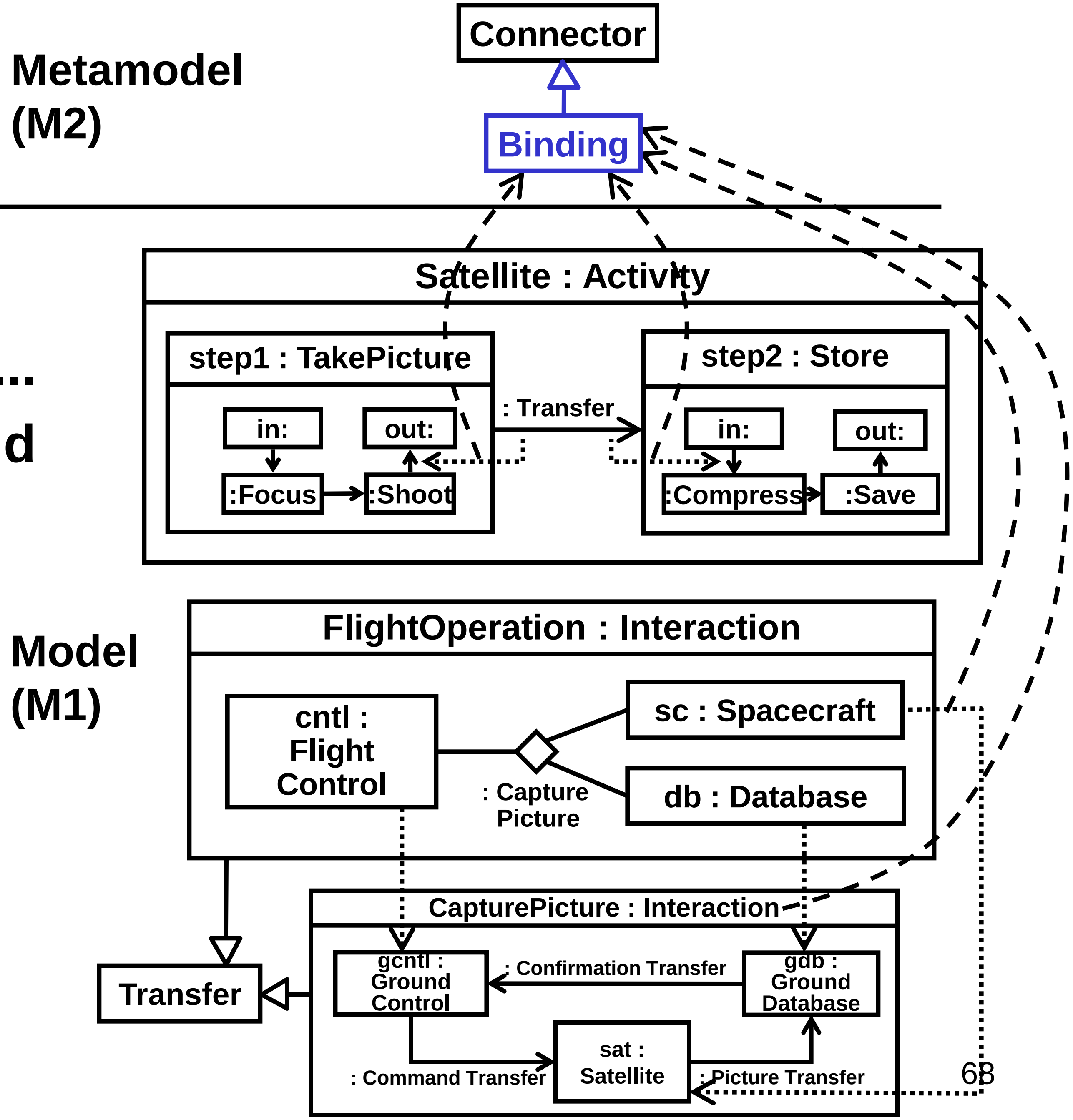
- Motivation
- Logical modeling
- Composite structure
- Behaviors as composites
  - Sequencing
    - Aside
  - Events
  - Participants
  - Flows (object flows and messaging)
    - External participants
    - Flow ordering
    - **Composition with flows and participants**

Composition with Flows and Participants

- Flows are part of behavior composition.
  - Activities have pins matching behavior parameters.
  - Interactions have arguments matching behavior parameters, used with collaboration, and collaboration role bindings.
- Requires specifying equivalence between transfers.

Bindings

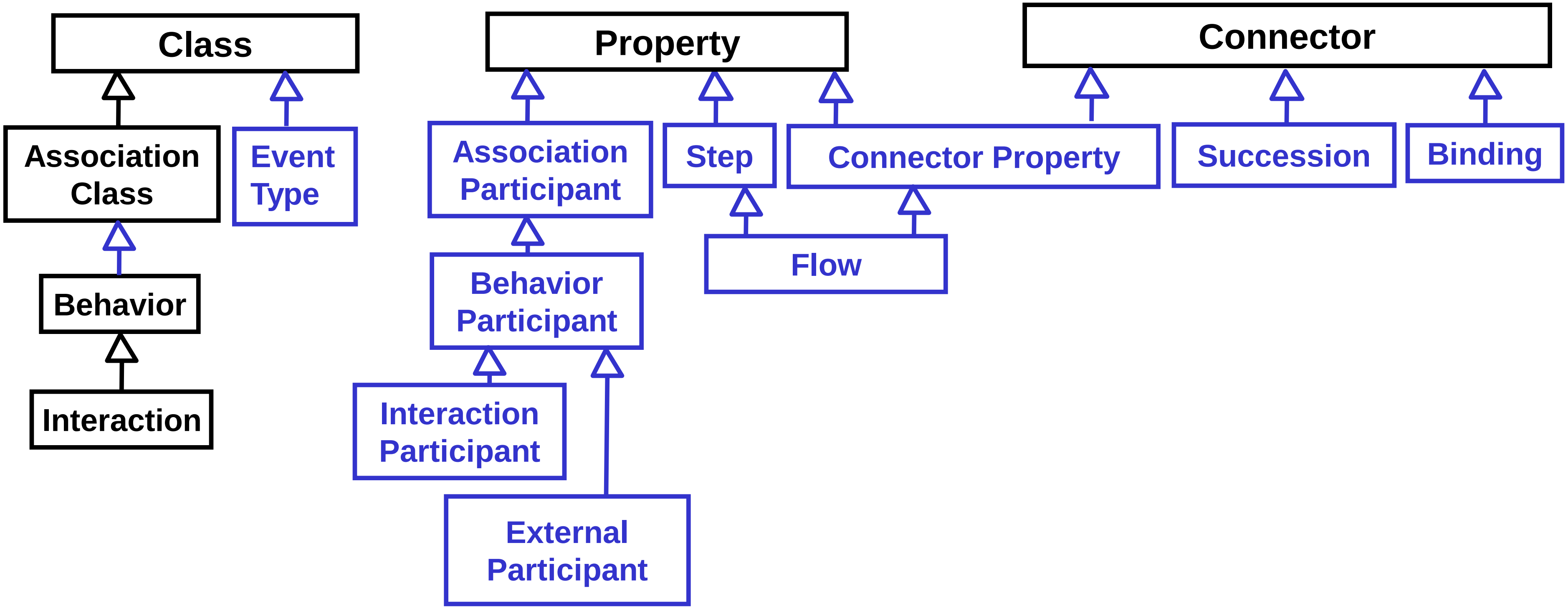
- Are connectors ....
- that link flows and participants at M1.
  - and specify equivalence of transfers at M0.



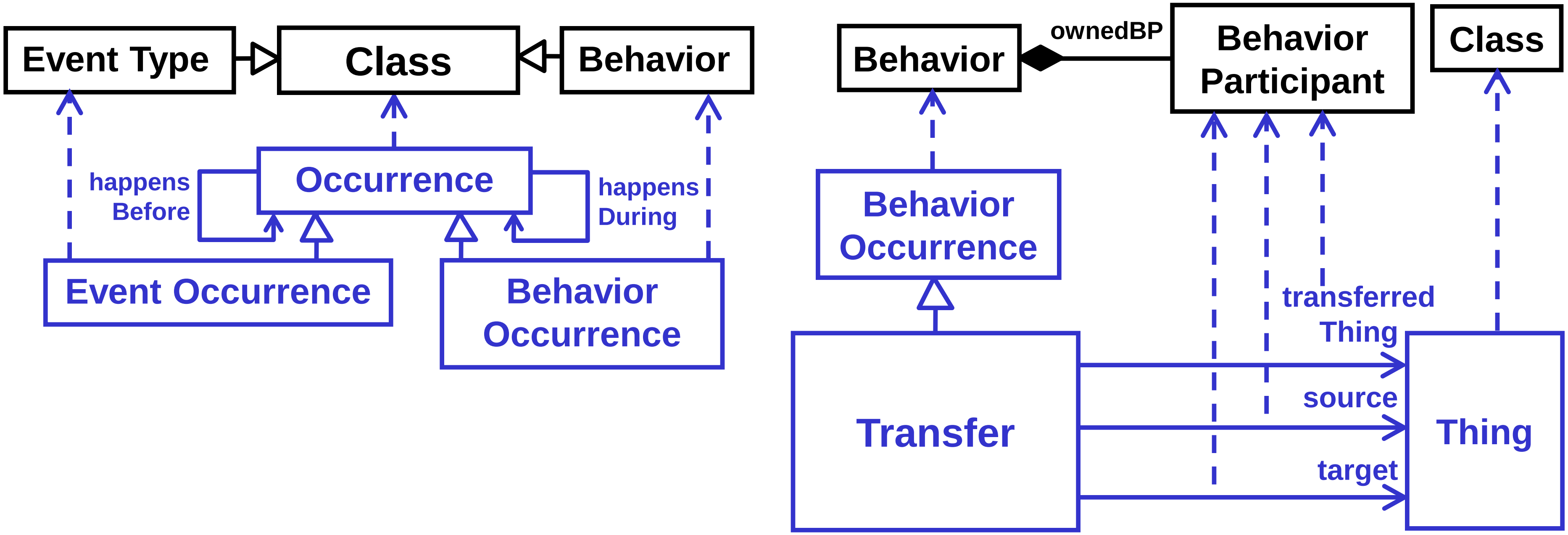
Summary

- Logical behavior modeling
  - Metamodel taxonomy
  - Model library
- Logical modeling

Metaclass Taxonomy



Model Library



Logical Modeling

- Semantics determines when M0 elements conform to M1 models.
- Metamodels should
  - reflect common semantics among M1 model elements.
  - have thin layers of clearly defined abstractions.
  - be augmented with M1 libraries to capture the relationship to M0.
- Behavior as example:
  - M1 behaviors and events specify M0 occurrences.
  - Specialize in metamodel from Class, Property, Connector, and Association Class.
  - Capture occurrences and temporal relations at M1.