State Machines as Composite Structure:

(Onto)Logical State Machines
Part 2

Bjorn Cole
Georgia Institute of Technology

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

Overview

- **RoadMap**
- **Motivation**
  - **Behavior, review**
  - **Interactions, review**
  - **State machines Part 1, review**
  - **State machines Part 2, requirements**
- **State Machines Solution, Part 2**
  1. **Objects reacting to stimuli**
  2. **Synchronizing state changes**
  3. **Managing stimuli**
- **Summary**

Behavior as Composite Structure Presentation Stack

Onto State Machines, Part 2

Onto State Machines, Part 1
(ad/18-12-09)

Onto Interactions
(ad/18-06-11)

Onto OO
(ad/18-09-07)

Onto Behavior Basics
(ad/2018-03-02)

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - State machines Part 1, review
  - State machines Part 2, requirements
- State Machines Solution, Part 2
  1. Objects reacting to stimuli
  2. Synchronizing state changes
  3. Managing stimuli
- Summary

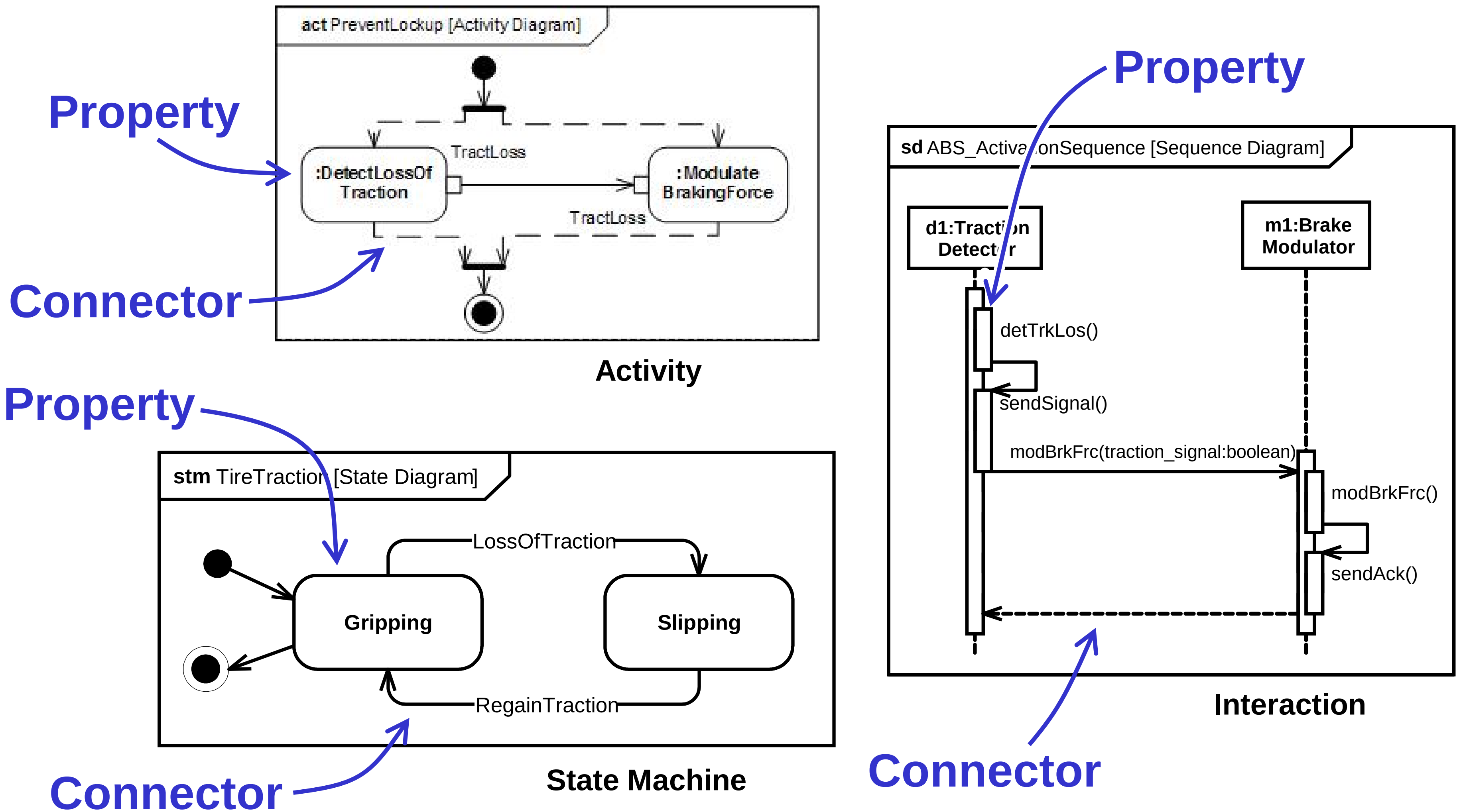
Original Problem

- 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

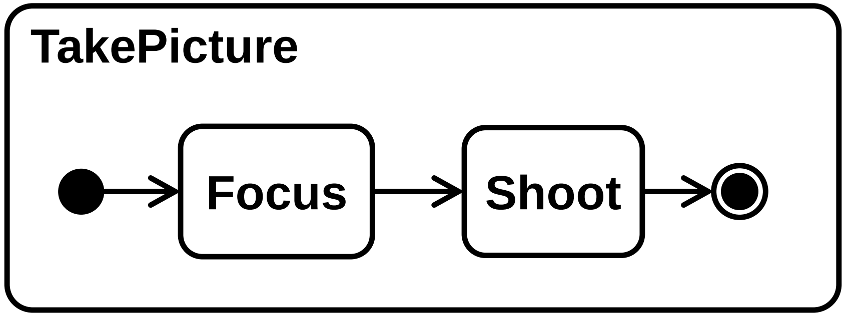
- Treat behaviors as assemblies of other behaviors.
  - Like objects are assemblies of other objects.
- Assembly = UML internal structure
  - Pieces represented by properties.
  - Put together by connectors.
- Put all behavior diagrams on the same underlying behavior assembly model.

Behaviors as Composite Structure

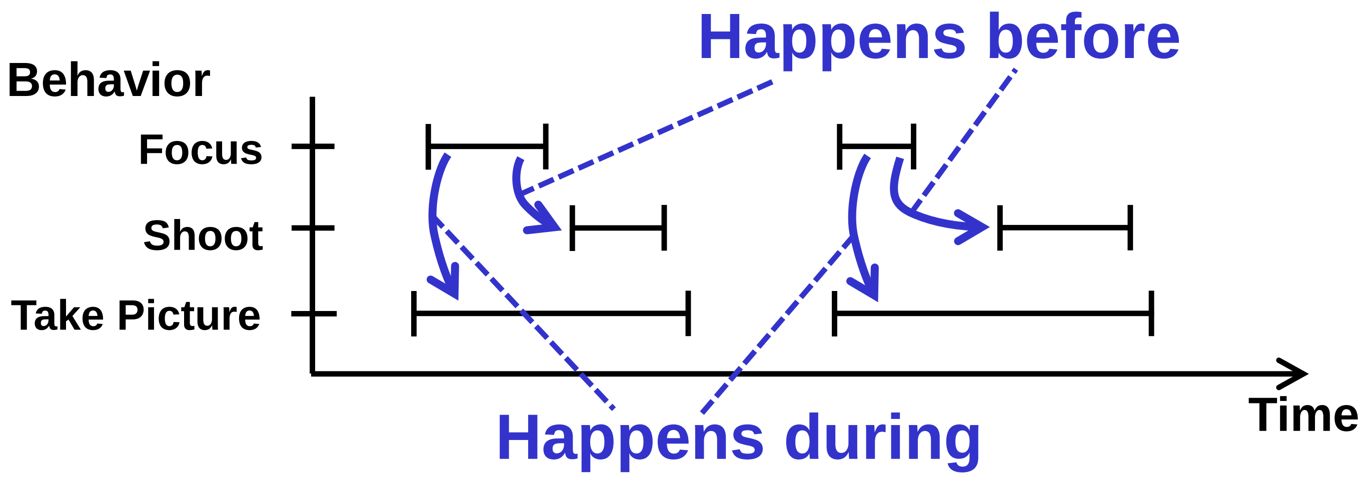


Behavior as Timing Constraints

Model
(M1)



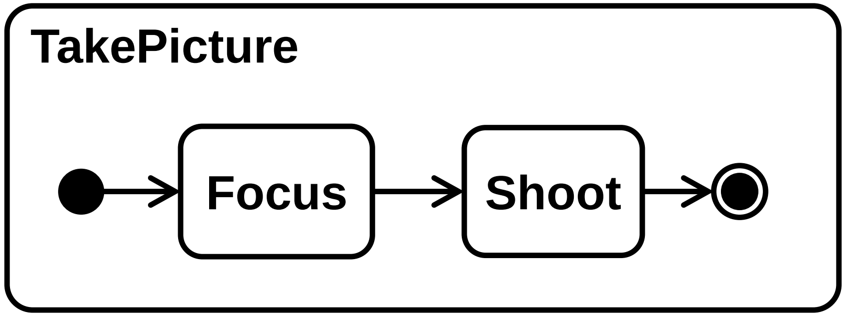
Things
Being
Modeled
(M0)



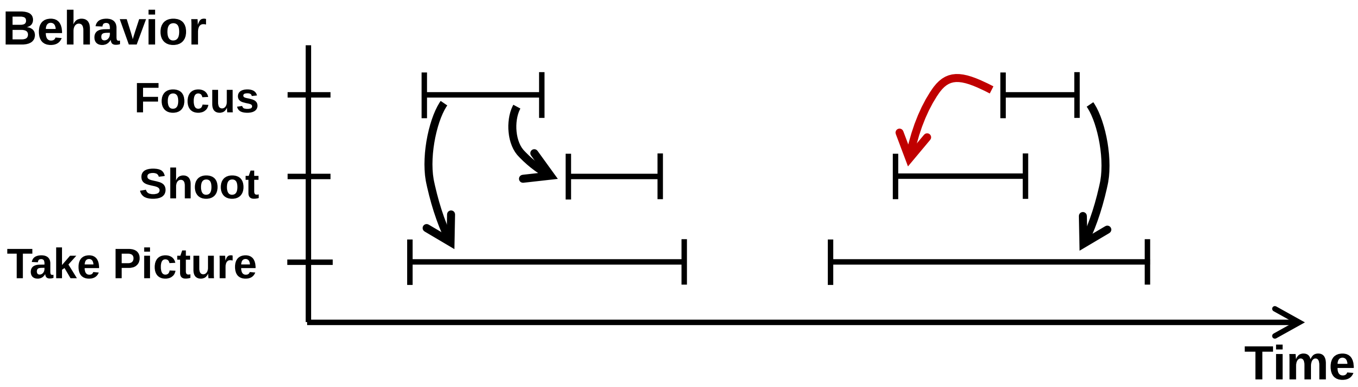
- **Behaviors model "things" happening over time.**
  - **With temporal relations (time constraints) between them.**

Behavior as Timing Constraints

Model
(M1)



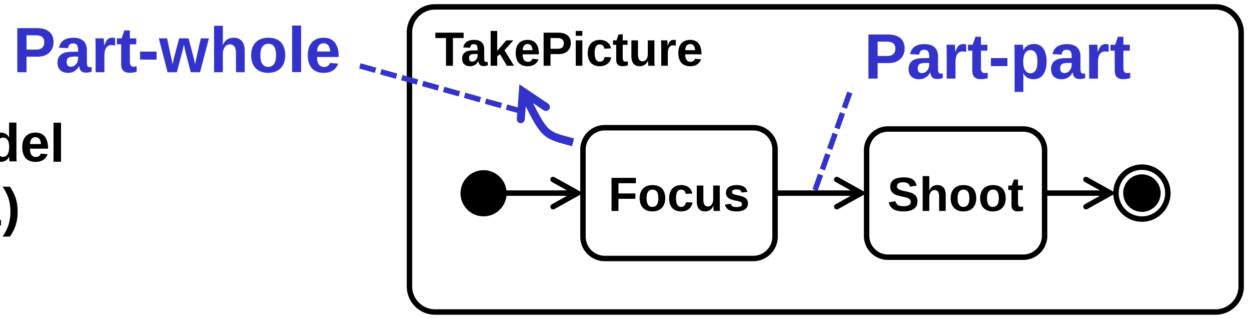
Things
Being
Modeled
(M0)



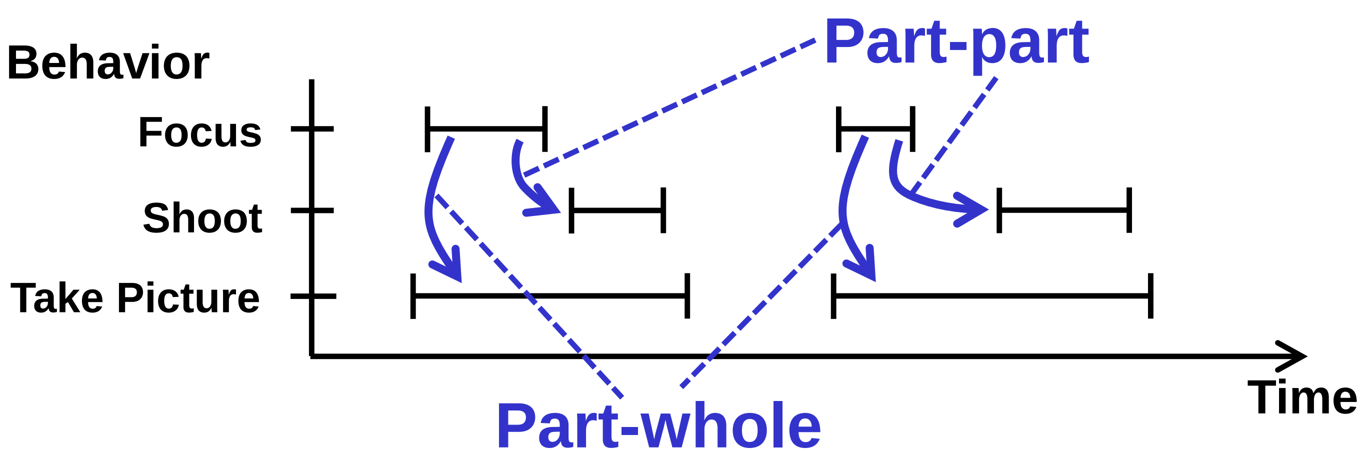
- **The TakePicture occurrence on the right does not follow the behavior model.**

Behavior as "Composite Timing"

Model (M1)

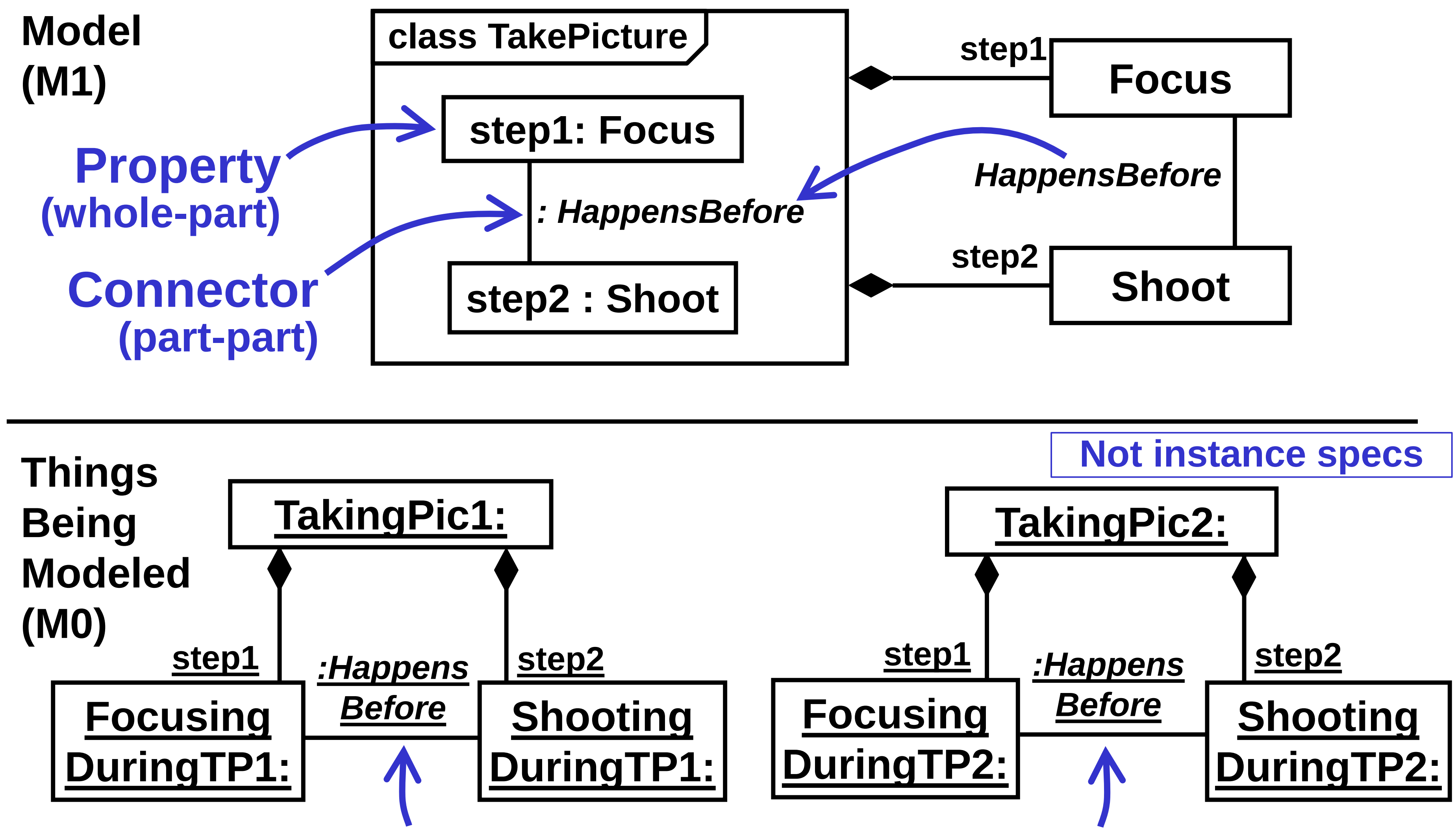


Things Being Modeled (M0)



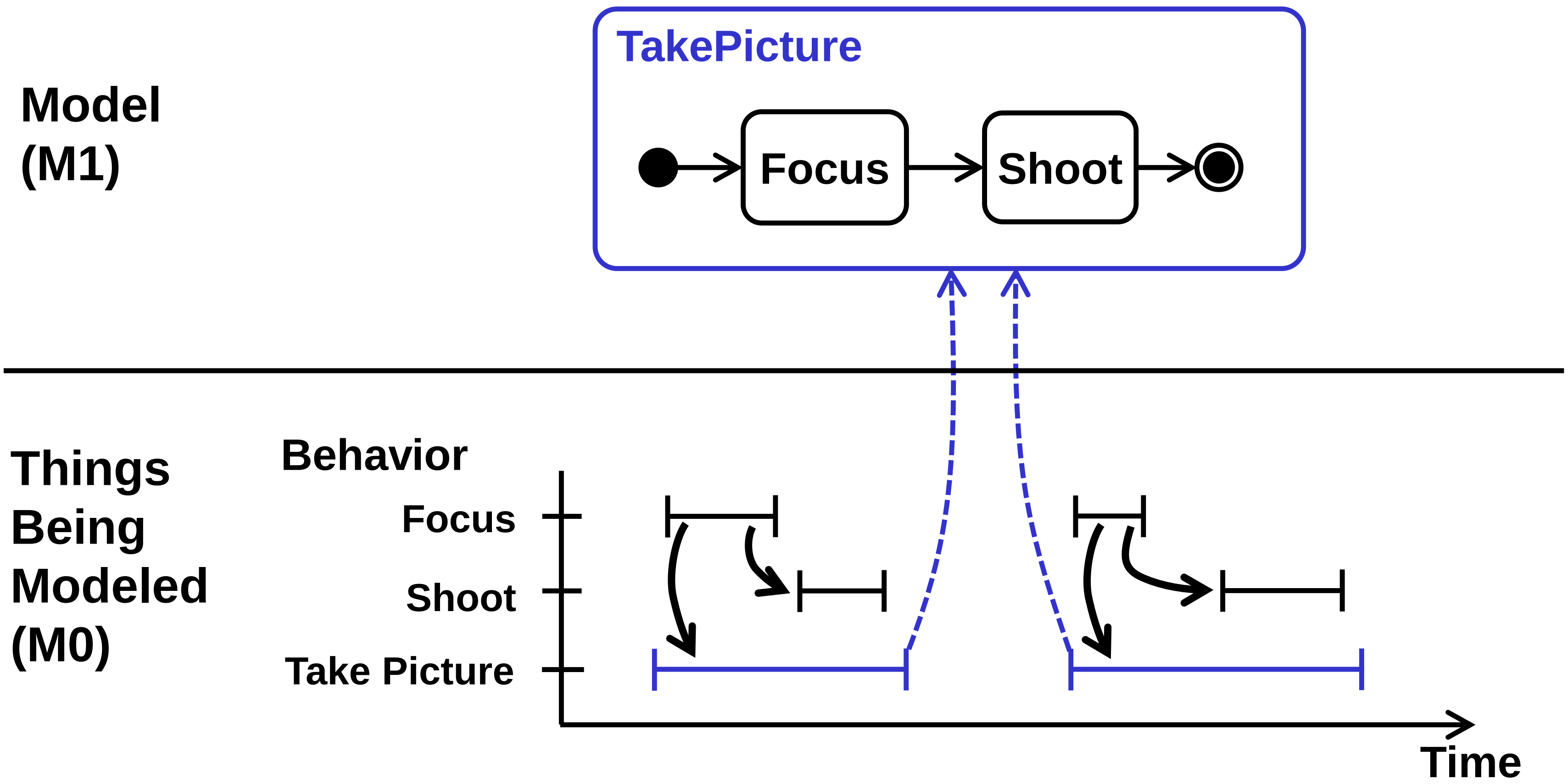
- **Composite structure relations are temporal:**
  - **Part-whole = happens during.**
  - **Part-part = happens before.**

Behavior as “Composite Timing”



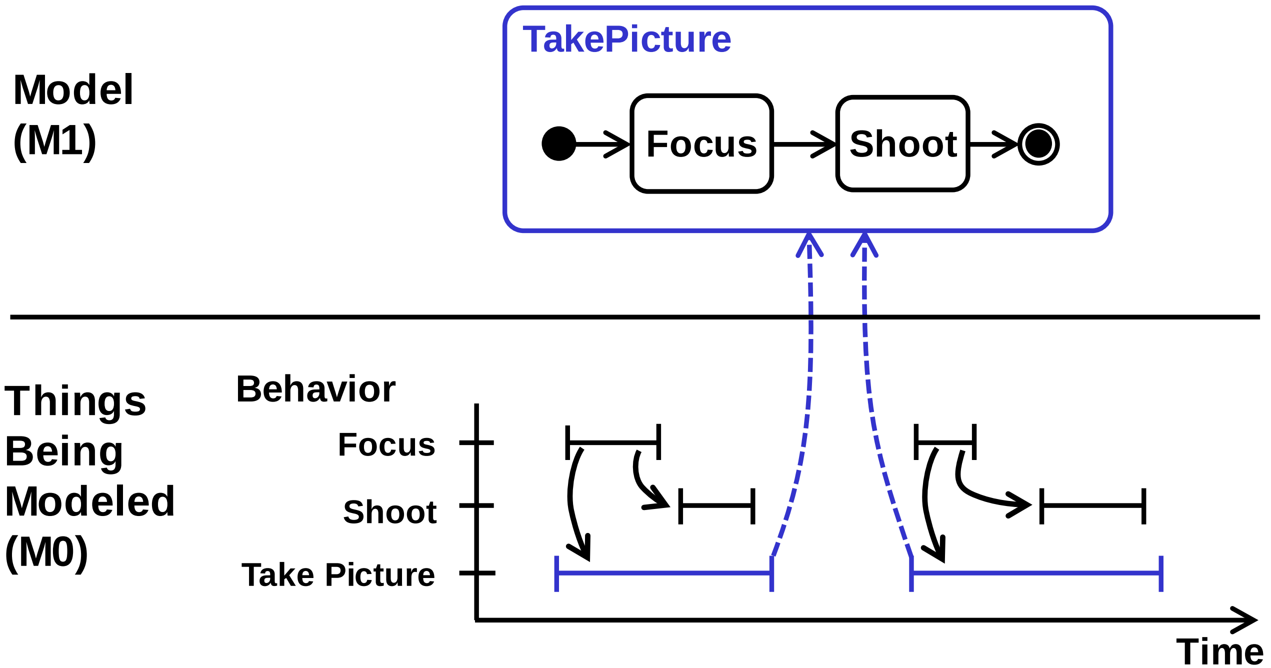
Focusing before shooting in same taking picture

Model and Things Being Modeled



- **Dashed arrows between M1 and M0 mean ....**

M0 → M1 Synonyms



Classified by

Modeled by

Specified by

Conforms to

Follows

Satisfies (logically)

Not quite: Instance of (in the OO sense)

Not *at all* : Execution of (in the software sense)

Behavior: What's Being Modeled?

Real,
Simulated,
or Desired
Things Being
Modeled (M0)

Not instance specs.

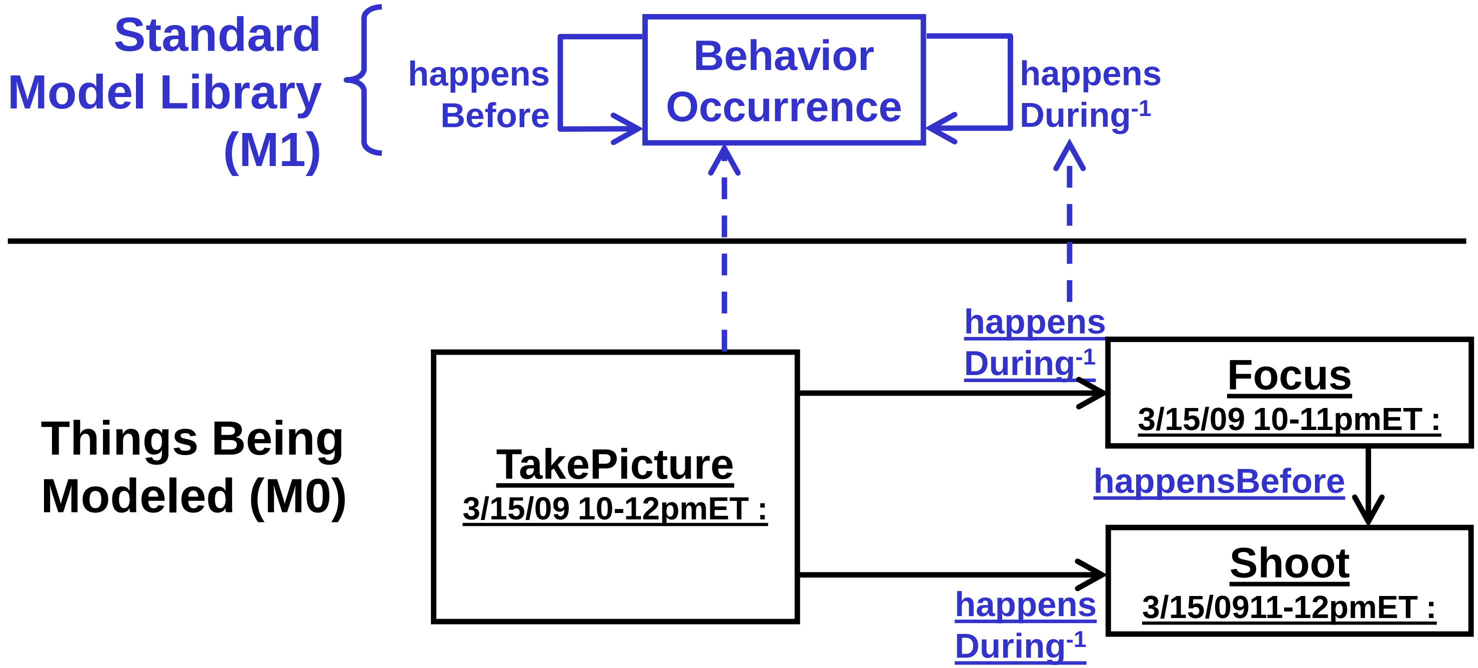
Focus
3/15/09 10-11pmET :

TakePicture
3/15/09 10-12pmET :

Shoot
3/15/0911-12pmET :

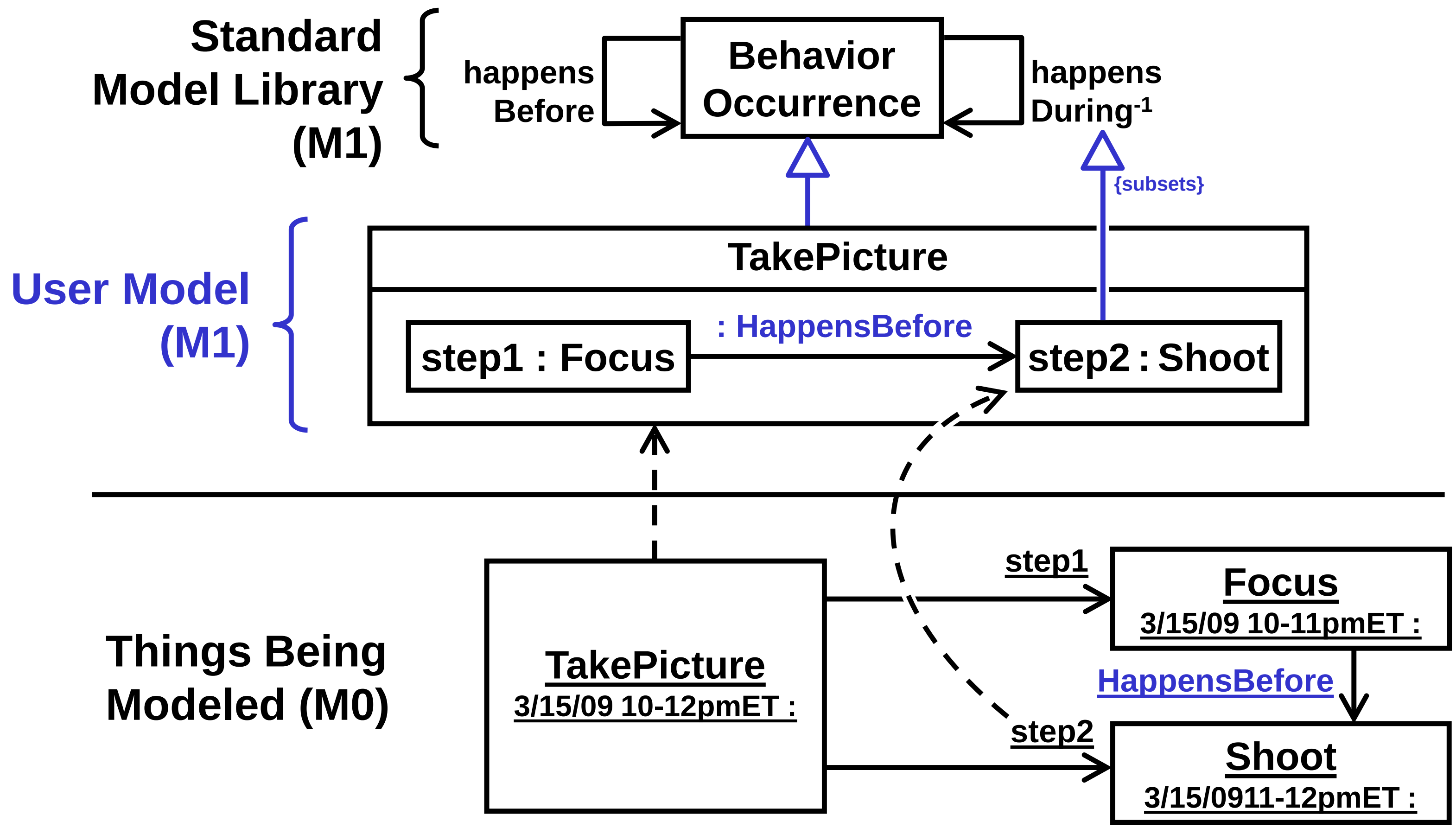
- "Things" that occur in time
  - Eg, taking a picture, focusing, etc.
  - Not "behaviors", "actions", etc.

Behavior: What's in Common?



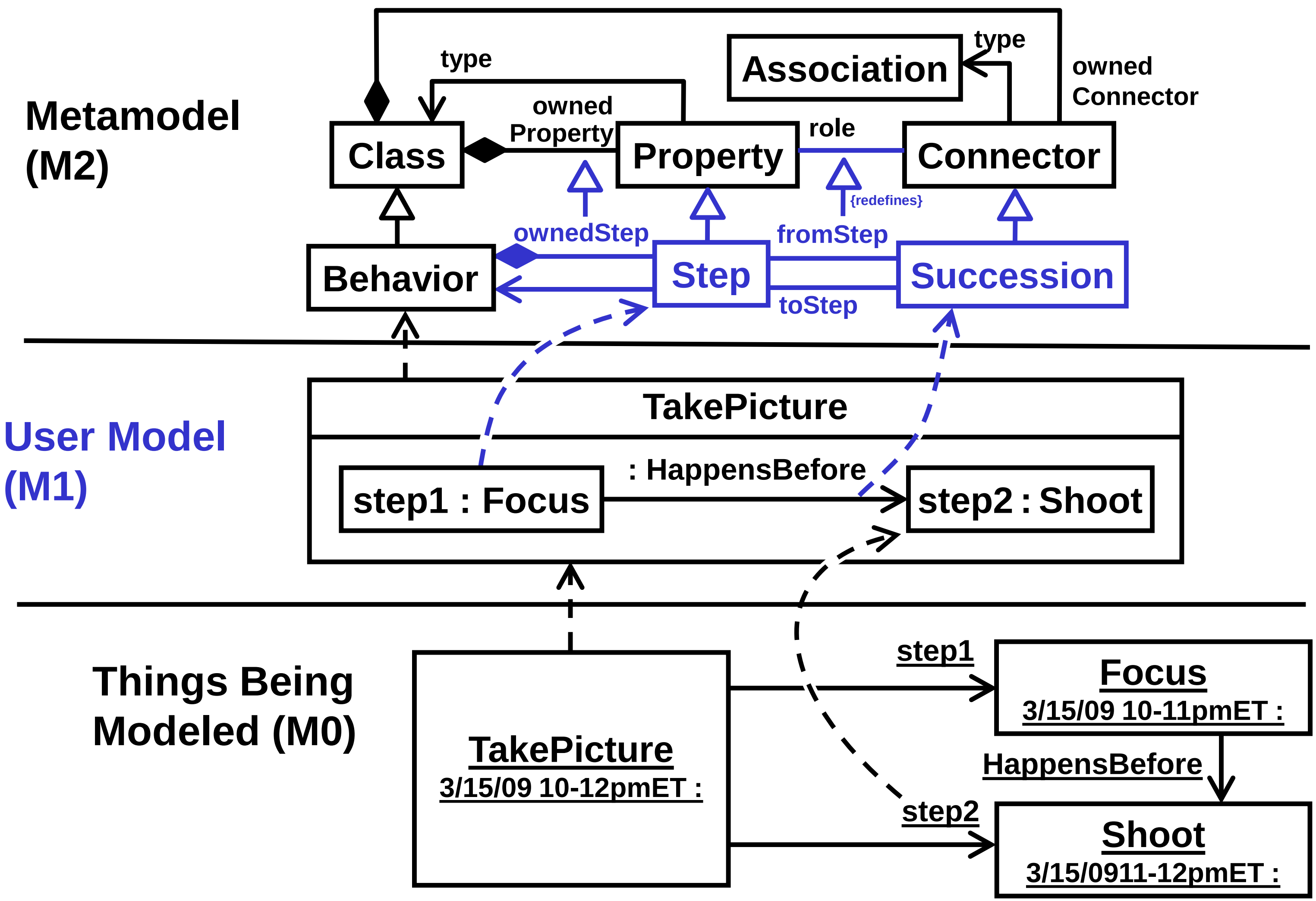
- **They happen before or during each other.**
  - **Construct M1 library for this.**
  - **Use it to classify things being modeled.**

Behavior: Use Library



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

Behavior: Too repetitive at M1?

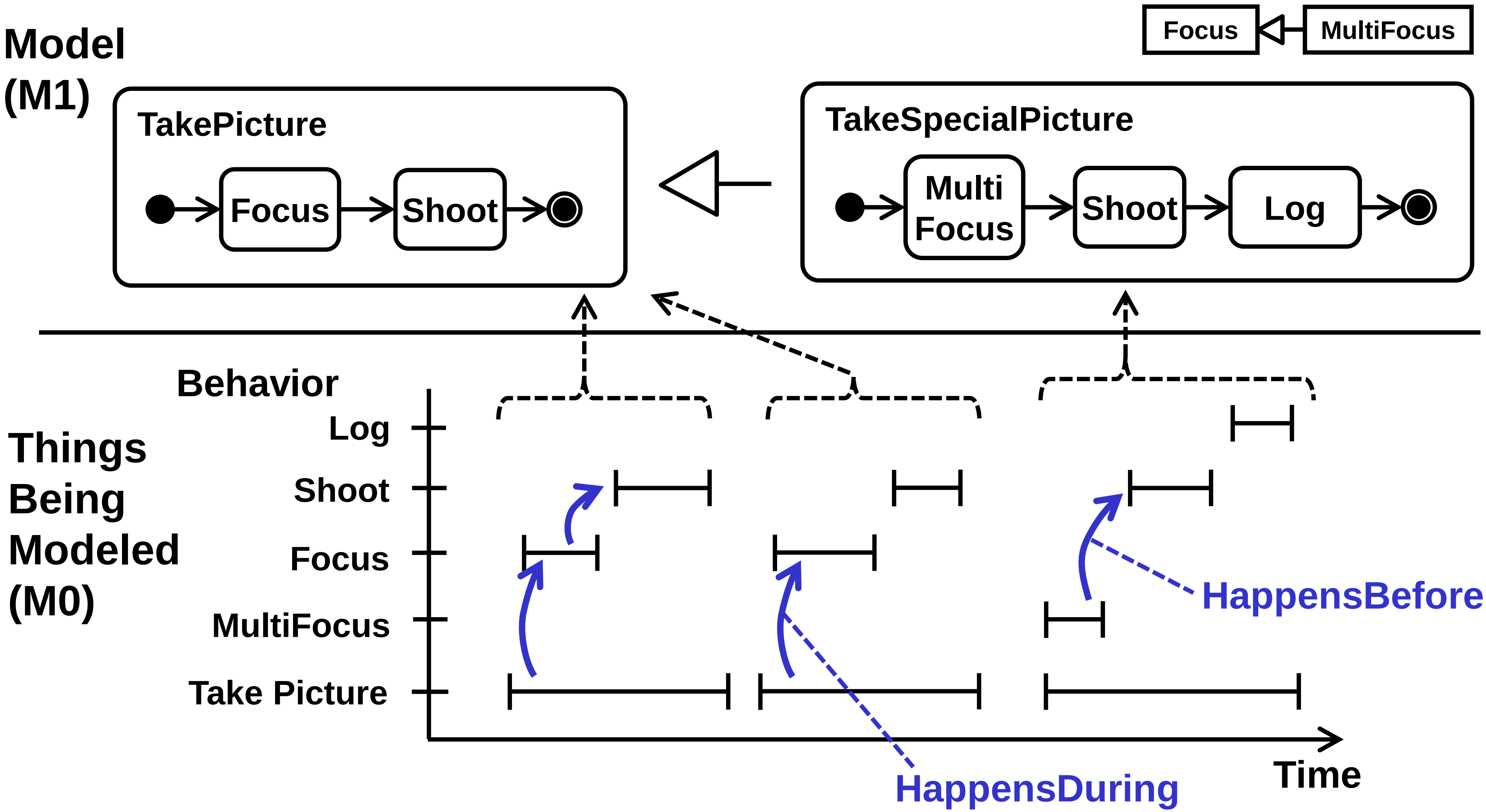


- Capture M1 patterns in M2 elements.
  - Tools apply patterns automatically.

Benefits: Original Problem

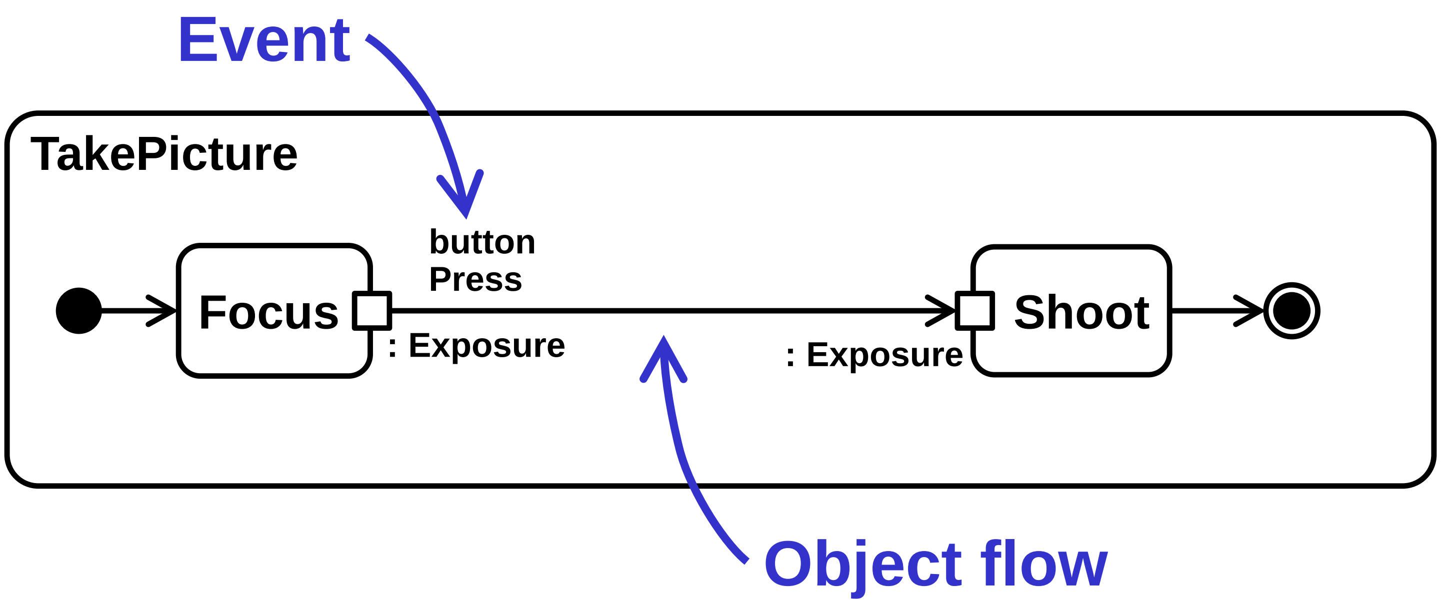
- **Flexibility in using metamodels**
  - **Add metaelements as needed to simplify library usage.**
- **Many metaelements become synonyms**
  - **Application / method / diagram-specific terminology sharing same semantics.**
  - **M2 actions, states, etc, => M1 happensDuring**
- **Learning UML and building analysis tools for it is easier**
  - **Due to shared semantics for variety of modeling language terminology.**

Benefits: Expressiveness



- Constraints are inherited in UML
  - including temporal constraints.

Benefits: Expressiveness



- **Combine activity and state machines.**
  - **States and actions happen during their "containing" occurrences, ordered in time.**

Benefits: Modeled Semantics

- **UML semantics is written in free text**
  - **Specifying an execution procedure for activities and state machines:**

Tokens are *offered* to an ActivityEdge by the source ActivityNode of the edge. Offers propagate through ActivityEdges and ControlNodes, according to the rules associated with ActivityEdges (see below) and each kind of ControlNode (see sub clause 15.3) until they reach an ObjectNode (for object tokens) or an ExecutableNode (for control tokens and some object tokens as specified by modelers, see ObjectNodes in sub clause 15.4). Each kind of ObjectNode (see sub clause
15.4) an
accepte
Activity
which a

The processing of Event occurrences by a StateMachine execution conforms to the general semantics defined in Clause 13. Upon creation, a StateMachine will perform its initialization during which it executes an initial compound transition prompted by the creation, after which it enters a *wait point*. In case of StateMachine Behaviors, a wait point is represented by a stable state configuration. It remains thus until an Event stored in its event pool is dispatched. This Event is evaluated and, if it matches a valid Trigger of the StateMachine and there is at least one enabled Transition that can be triggered by that Event occurrence, a single StateMachine *step* is executed. A step involves executing a compound transition and terminating on a stable state configuration (i.e., the next wait point). This cycle then repeats until either the StateMachine completes its Behavior or until it is asynchronously terminated by some external agent.

  - **and trace classification in interactions:**

Clause 13, Common Behaviors, describes the general semantics of the execution of Behaviors. Interactions are kinds of Behaviors that model emergent behaviors, as defined in sub clause 13.1. As discussed in sub clause 13.2.3, the execution of a Behavior results in an execution trace. Such a trace is a sequence of event occurrences, which, in this clause, will be denoted <e1, e2, . . . , en>. Each event occurrence may also include information about the values of all relevant objects at the point of time of its occurrence.
The semantics of an Interaction are expressed in terms of a pair [P, I], where P is the set of valid traces and I is the set of invalid traces. P ! I need not be the whole universe of traces. Two Interactions are equivalent if their pairs of trace-sets are equal. The semantics of each construct of an Interaction (such as the various kinds of CombinedFragments) are

- **Model in standard libraries.**

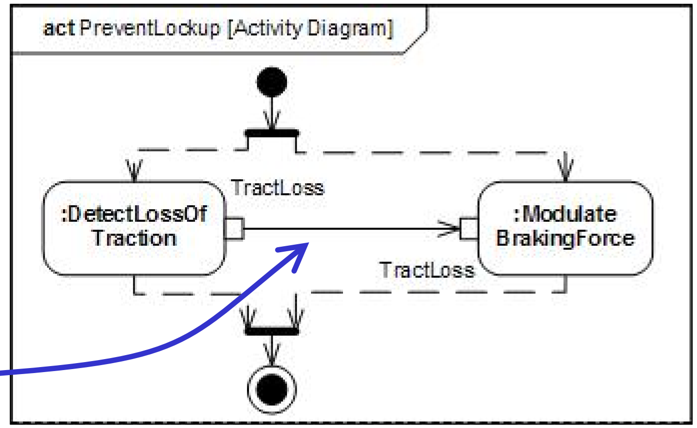
Benefits: Classification Semantics

- Standard execution models for UML
  - fUML, PSCS, PSSM
  - Procedures that create a behavior occurrence
    - Conforming to a UML model.
  - Don’t tell whether
    - An existing behavior occurrence conforms.
    - Tools are producing correct occurrences
- Classification does is the opposite
  - Tells whether an existing behavior occurrence conforms to a model.
  - Doesn’t say how to create an occurrence.
  - Execution engines are constraint solvers.

Overview

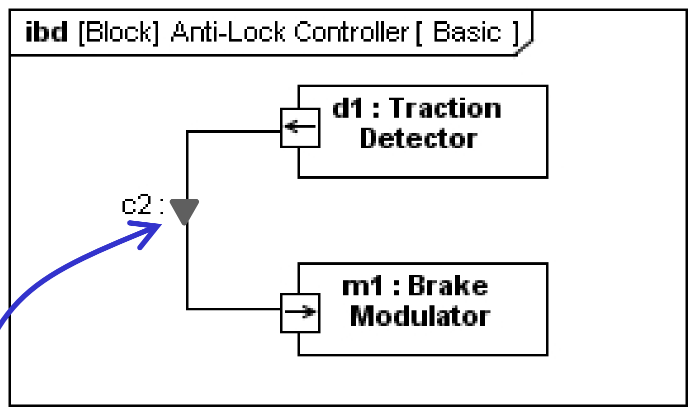
- RoadMap
- Motivation
  - Behavior, review
  - **Interactions, review**
  - State machines Part 1, review
  - State machines Part 2, requirements
- State Machines Solution, Part 2
  1. Objects reacting to stimuli
  2. Synchronizing state changes
  3. Managing stimuli
- Summary

Interactions Problem



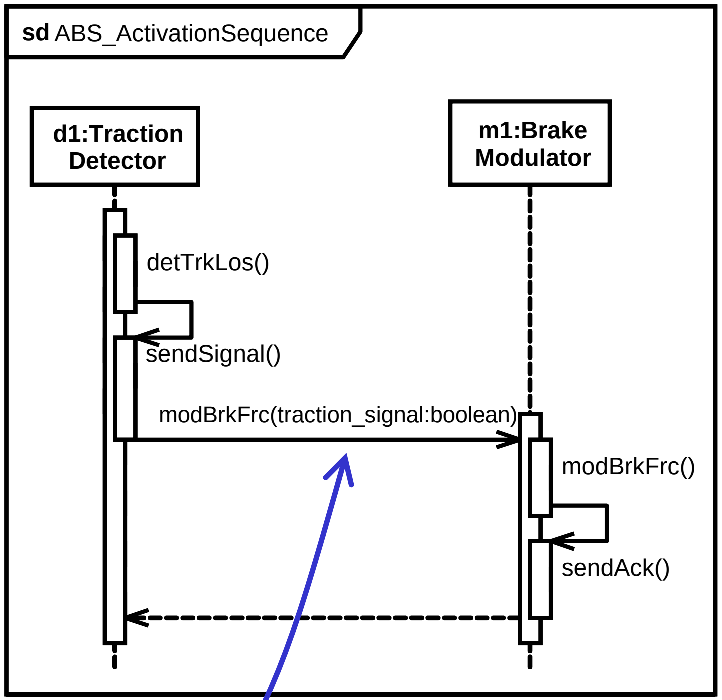
Object Flow

Activity



Item Flow

SysML Internal Block Diagram



Interaction

Message

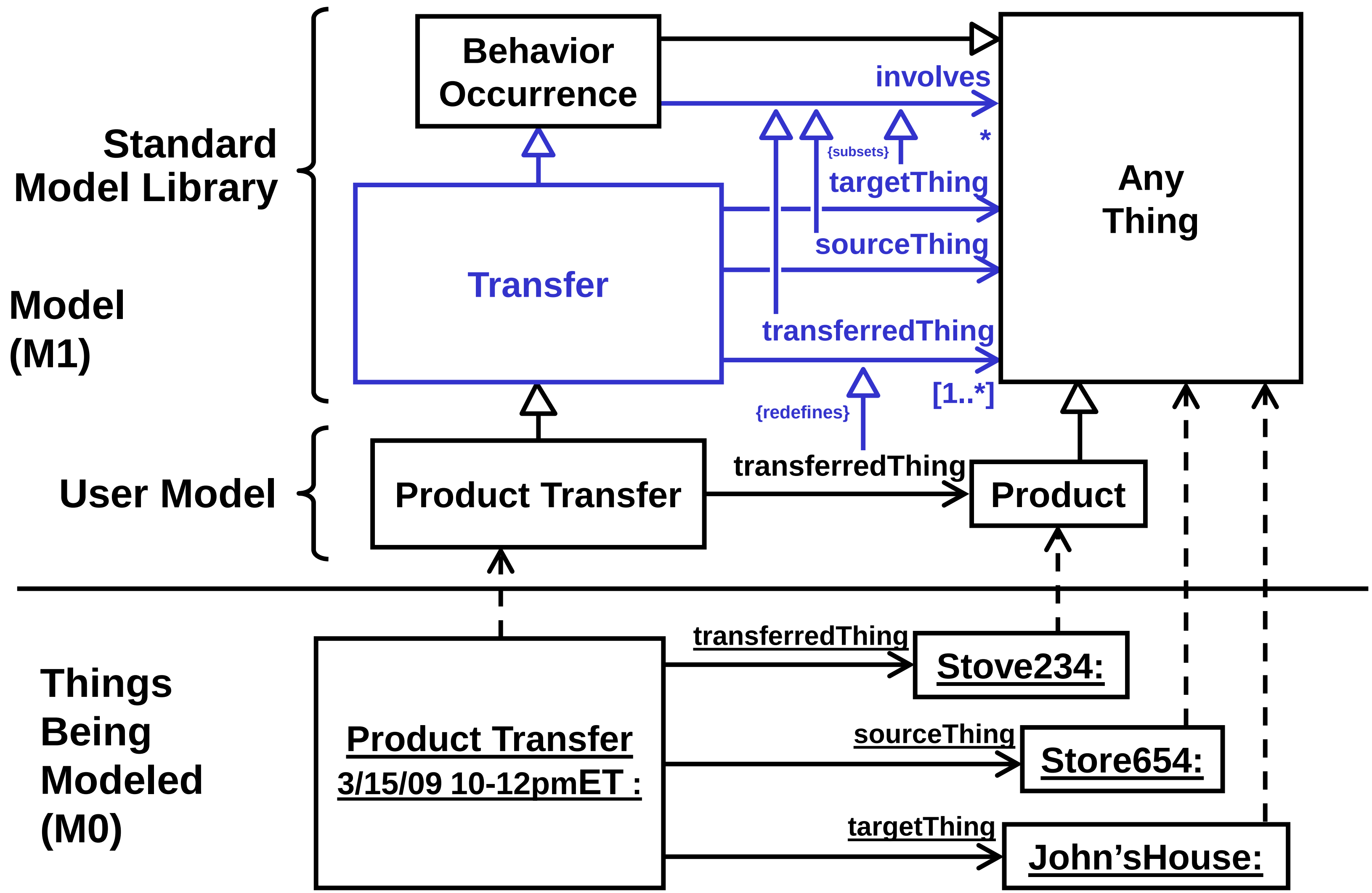
Interactions Requirements

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.

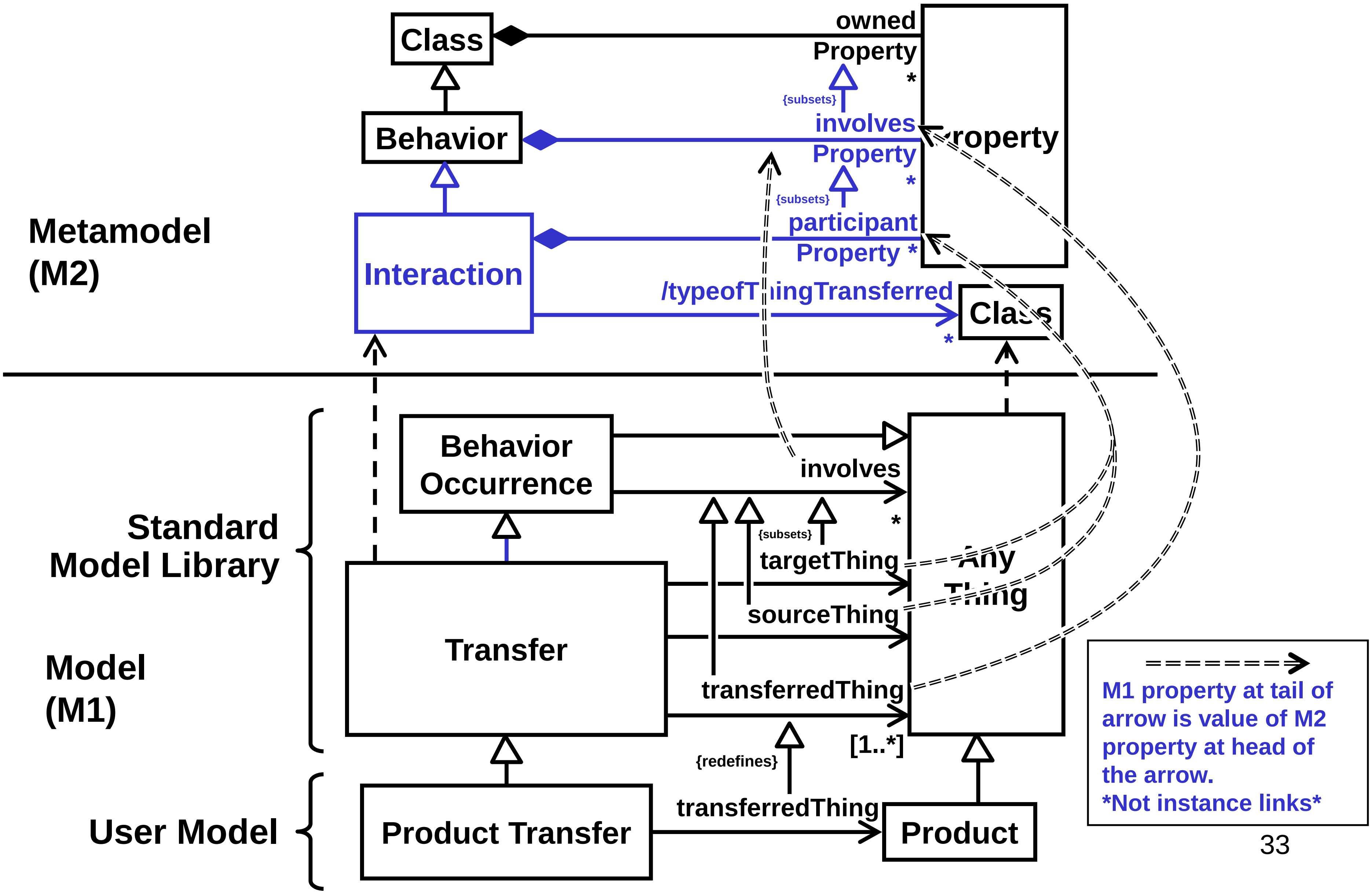
Interactions Solution (Part 1)
(between things that outlive interactions)

- Flows **happen in time**.
  – They are behaviors.
- Start when an entity **begins flowing**.
  – Leaves output pin of an action.
    … execution on a lifeline.
    … SysML out flow property.
- End when the entity **stops flowing**.
  – Arrives at input pin of an action.
    … execution on a lifeline.
    … SysML in flow property.

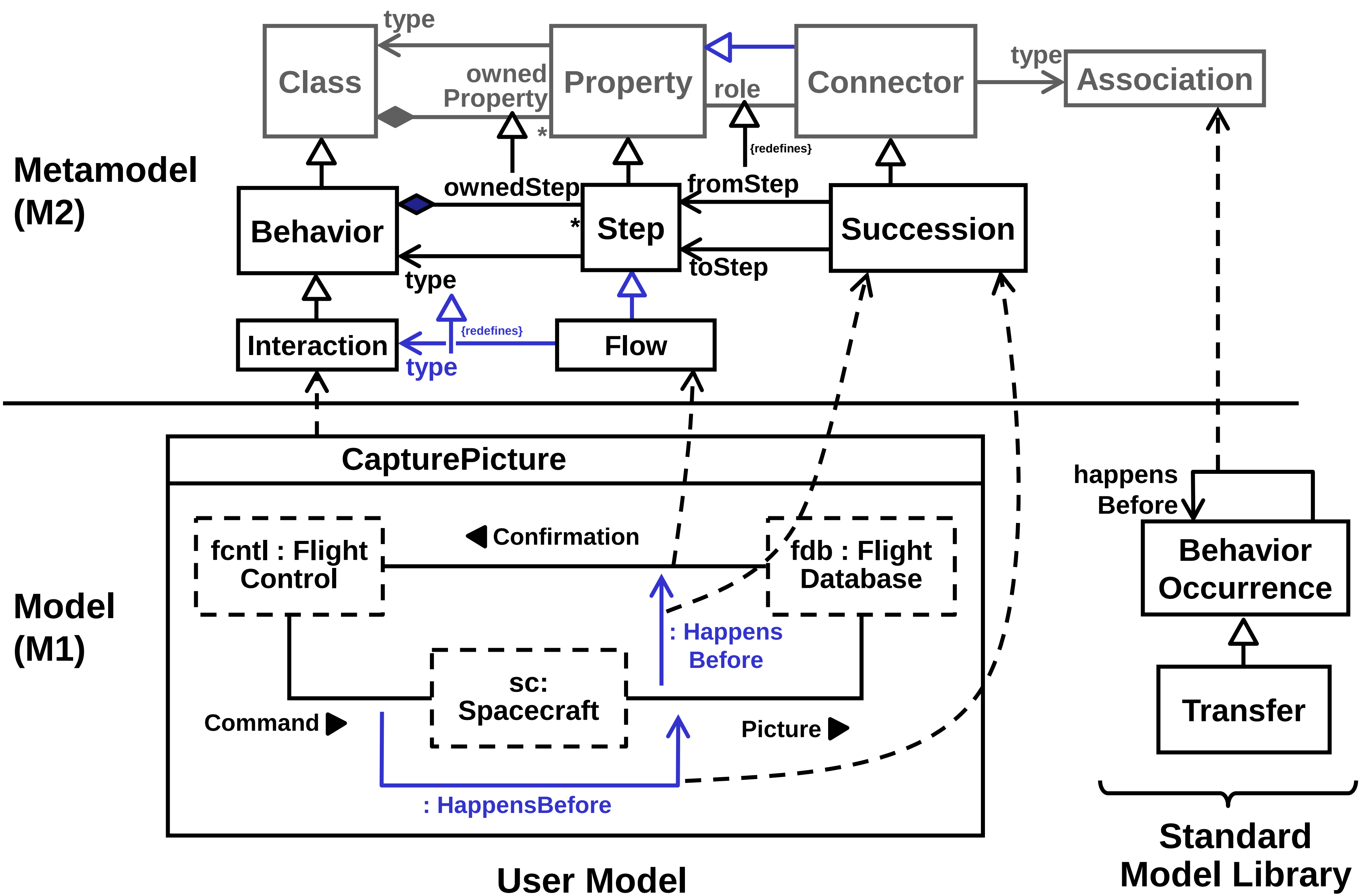
Transfers (M1)



Interactions (M2)



Flow Steps


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

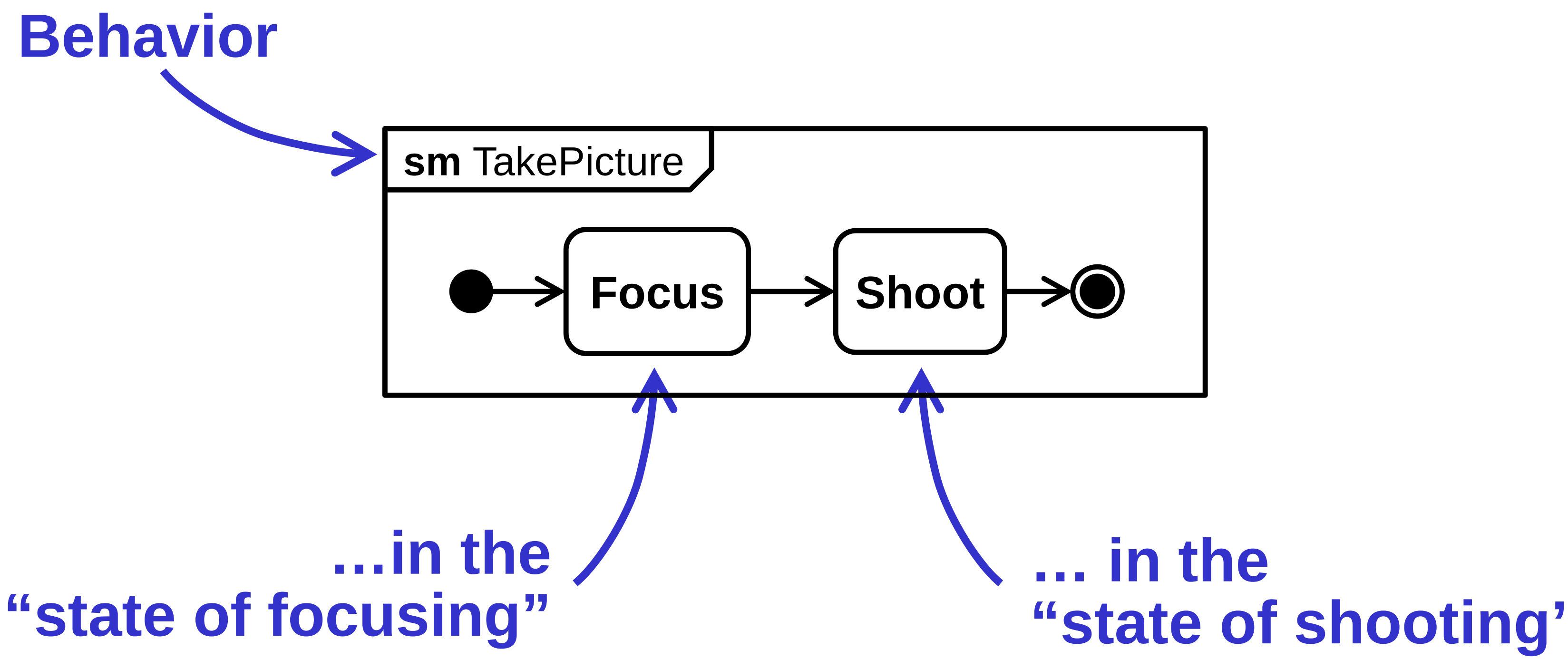
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - **State machines Part 1, review**
  - State machines Part 2, requirements
- State Machines Solution, Part 2
  1. Objects reacting to stimuli
  2. Synchronizing state changes
  3. Managing stimuli
- Summary

States of What?

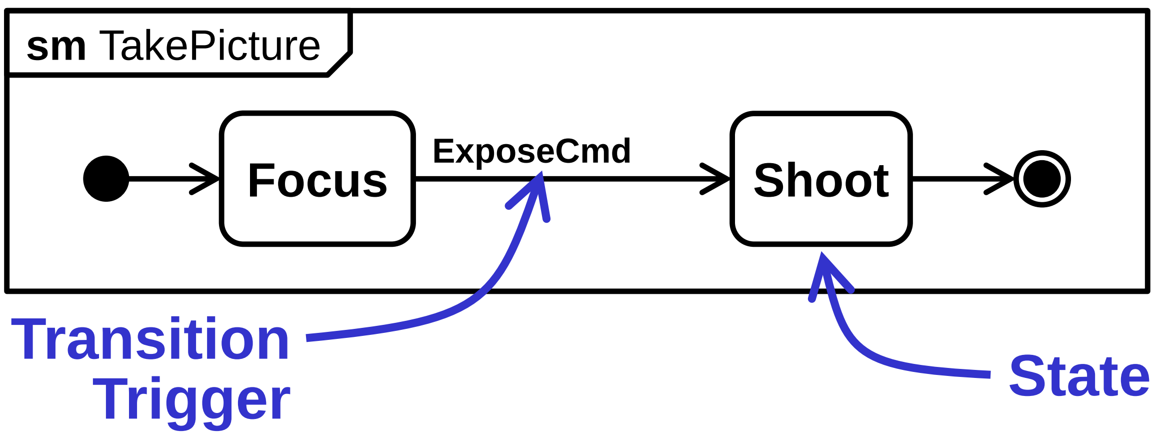
- Objects, based on properties
  - Person in married state = has a spouse.
- Behaviors, based on past behavior
  - Vending machine in dispensing state after receiving selection and money states.
  - *UML states are mostly of behaviors* ...
    - … tied to objects.
    - Weakly include object state invariants.
- Both kinds can be in “machines” that react to external stimuli.

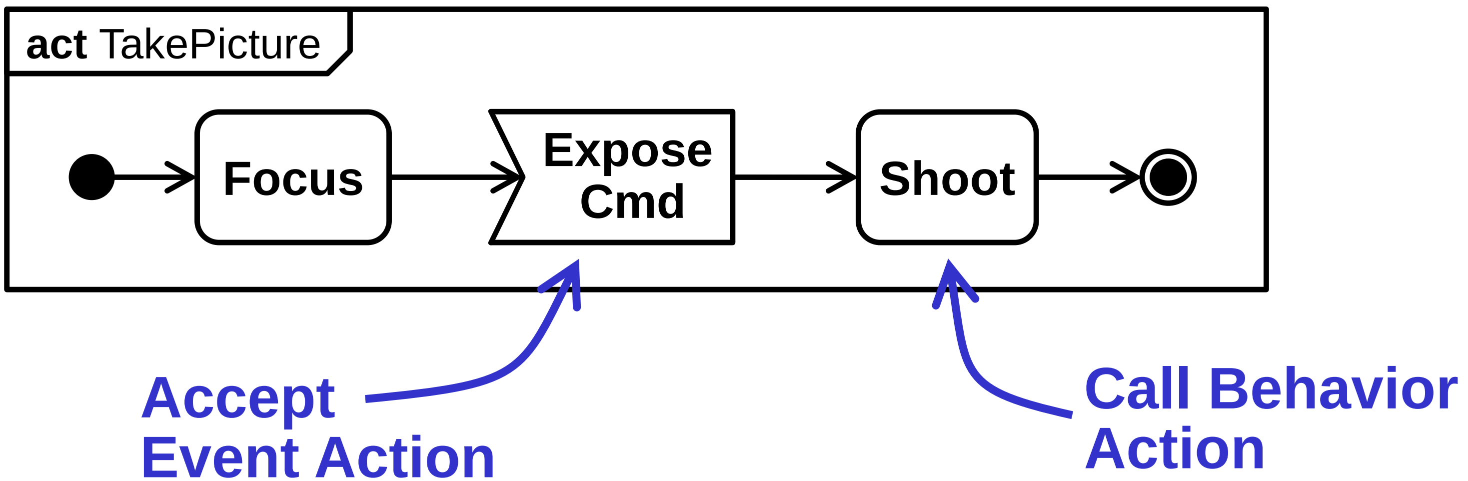
States of Behaviors



- States of behaviors = steps in behaviors
  – Properties typed by other behaviors.
- Part 1 assumed stimuli arrived directly at behaviors.

State Machine Problem, Part 1





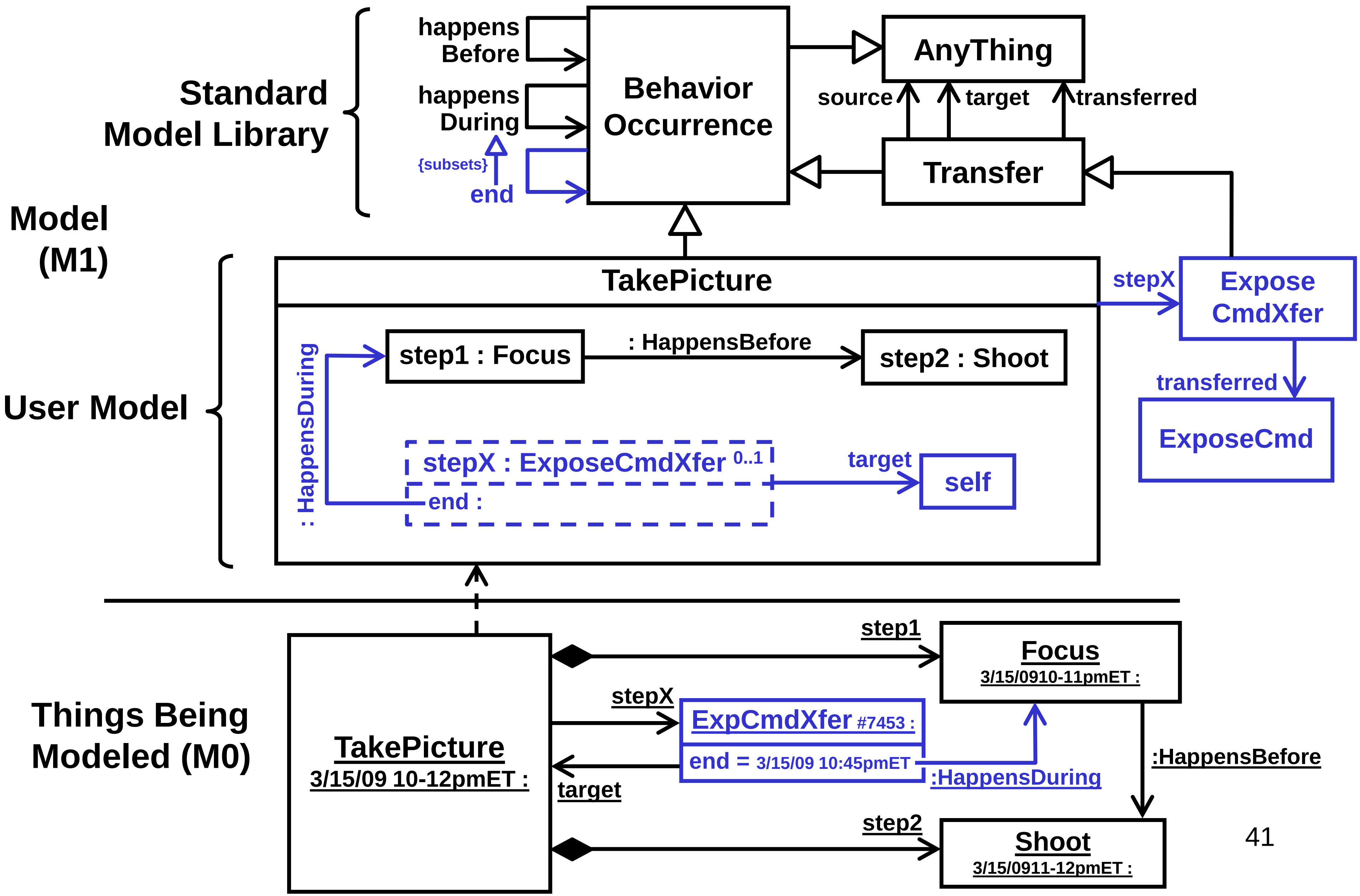
State Machine Requirements, P1

1. Must selectively react to stimuli (“events”).
   – Based on kind of stimulus and …
   – … current & previous stimuli/reactions (“states”)
2. Must simplify reaction behaviors, splitting them up …
   – by state and between states (transitions).
   – within states.
3. Must react to past events
   – Can have complicated reaction rules to events in the past.

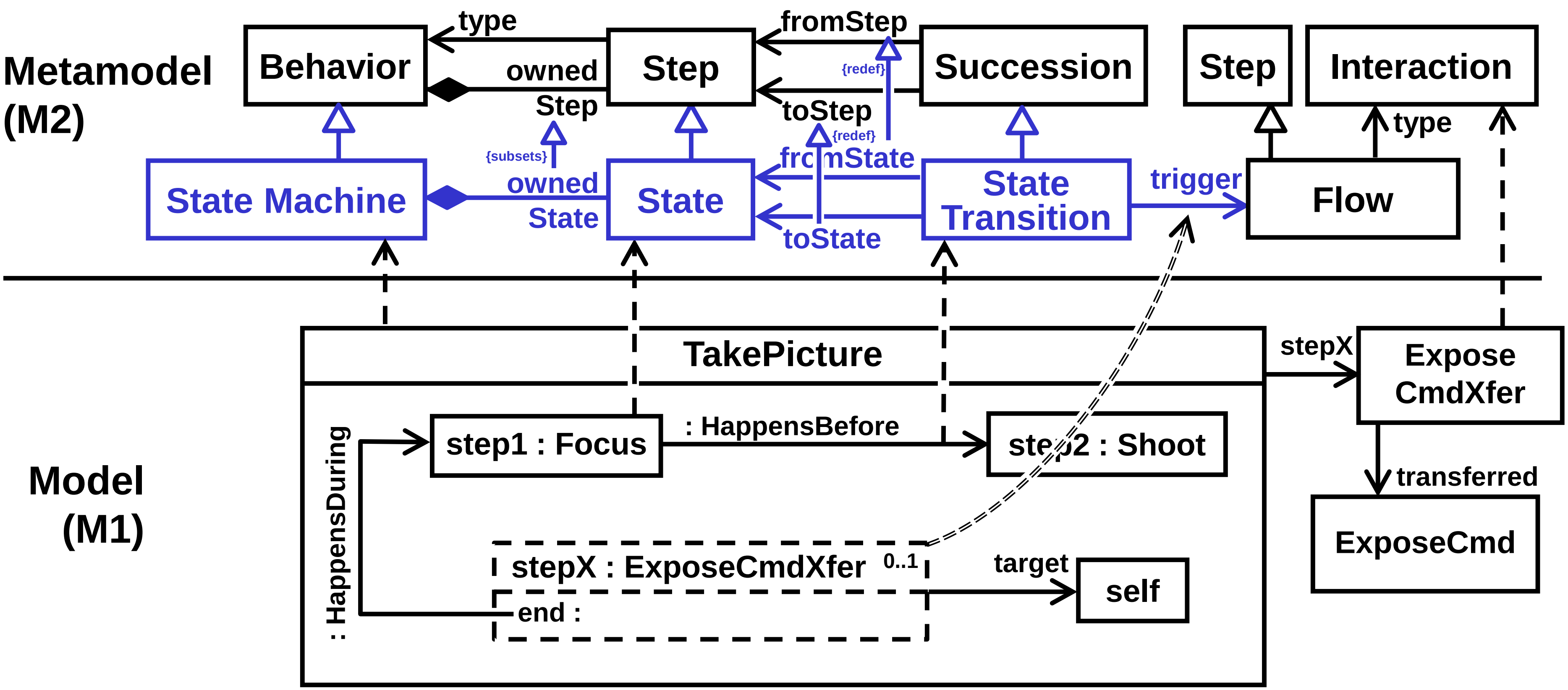
State Machine Solution (Part 1.1)
(Reacting to stimuli)

- UML events = things "arriving" at objects
  - Signals, operation calls
  - Not events happening externally
    - Except unmodeled "changes" to anything.
- Treat as ends of transfers targeting objects.
  - Receiver doesn't specify sender.

UML Events = Ends of Transfers



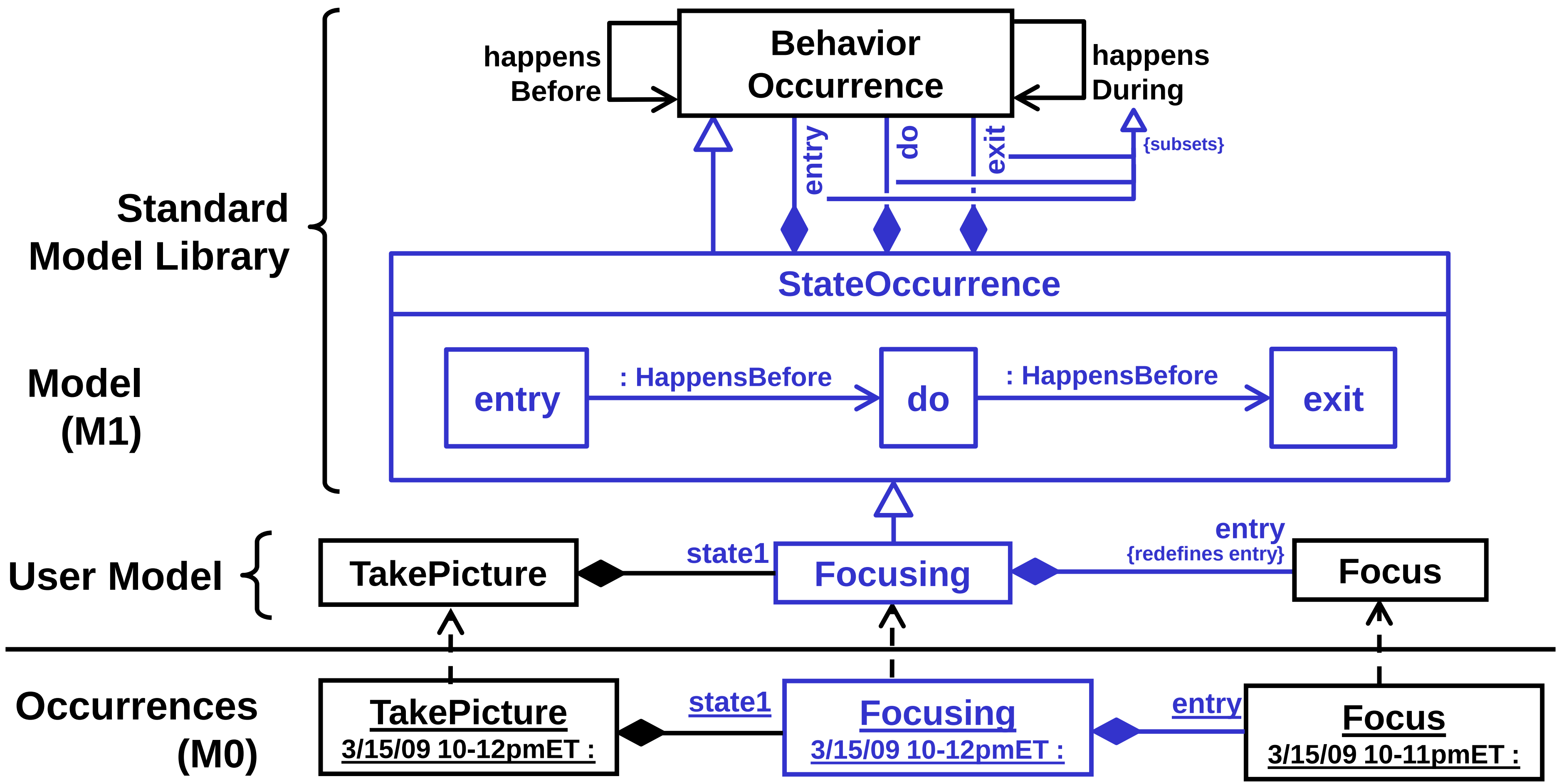
State Machines (M2)



- Transitions are successions that …
  - go out of steps …
  - that identify interactions (triggers) …
  - that end during and target the machine.

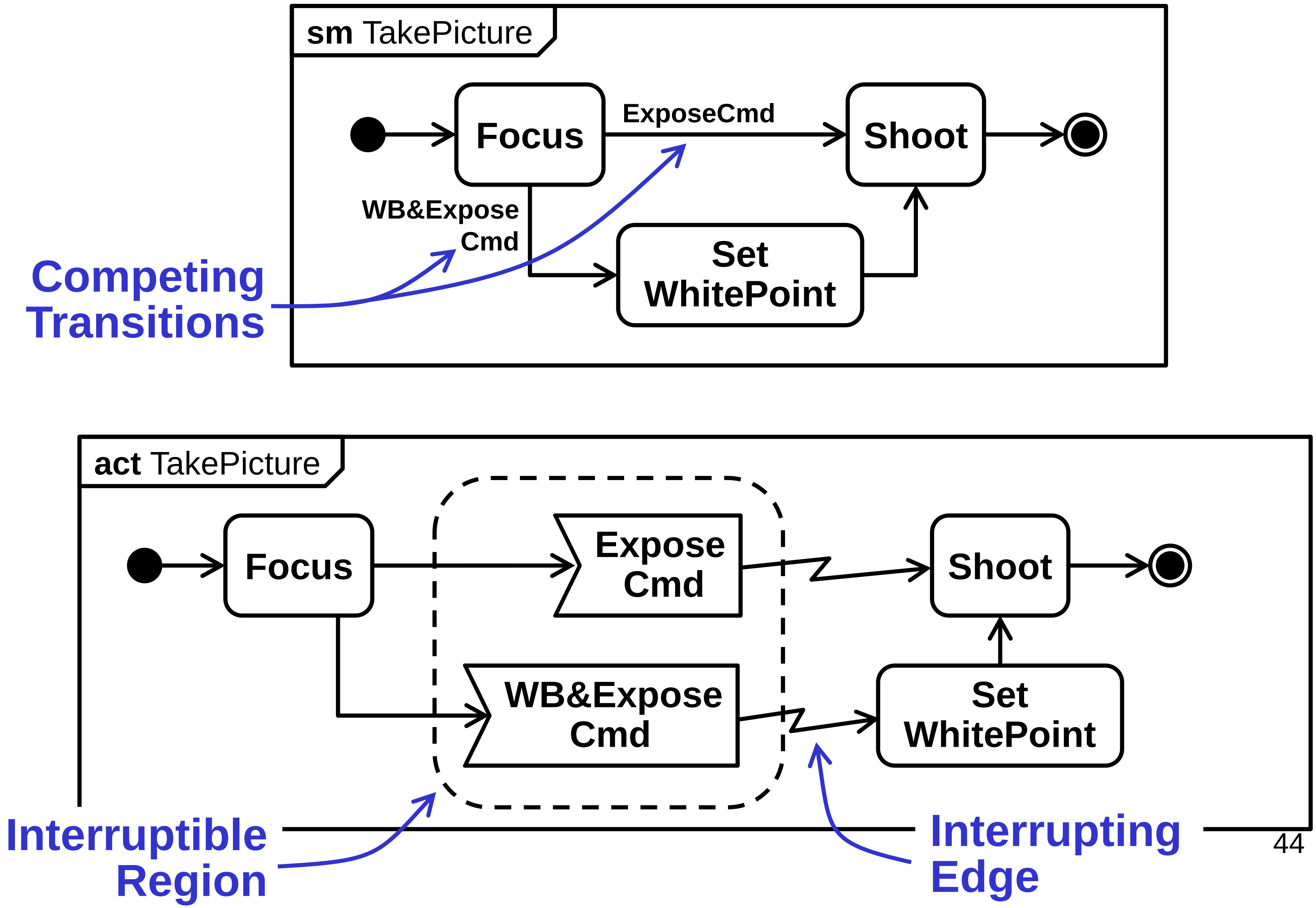
M1 property at tail of arrow is value of M2 property at head of the arrow.
Not instance links

State Behaviors (M1)

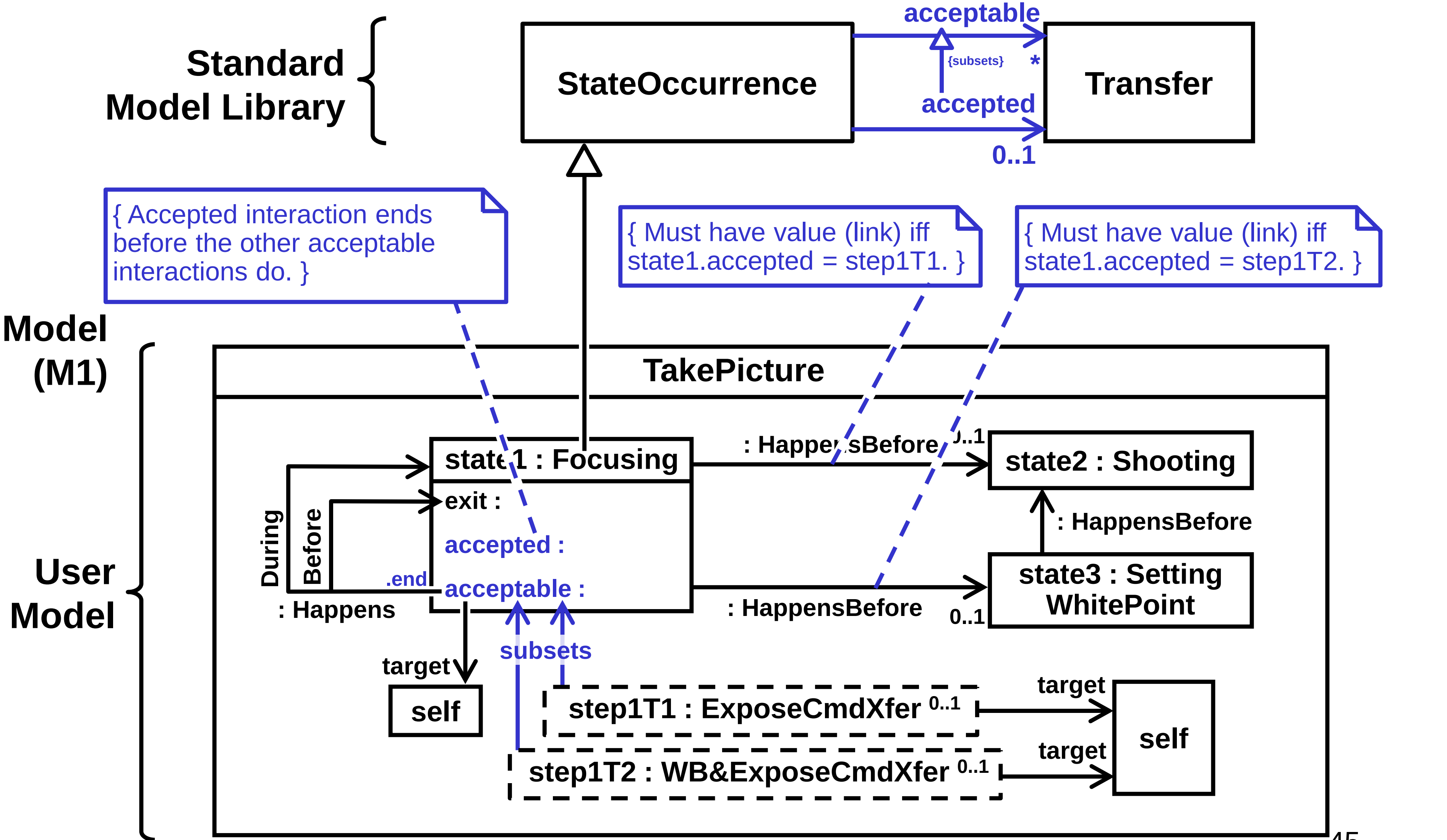


- **State occurrences**:
  - Are behavior occurrences typing state properties...
  - with exactly three step properties ordered in time

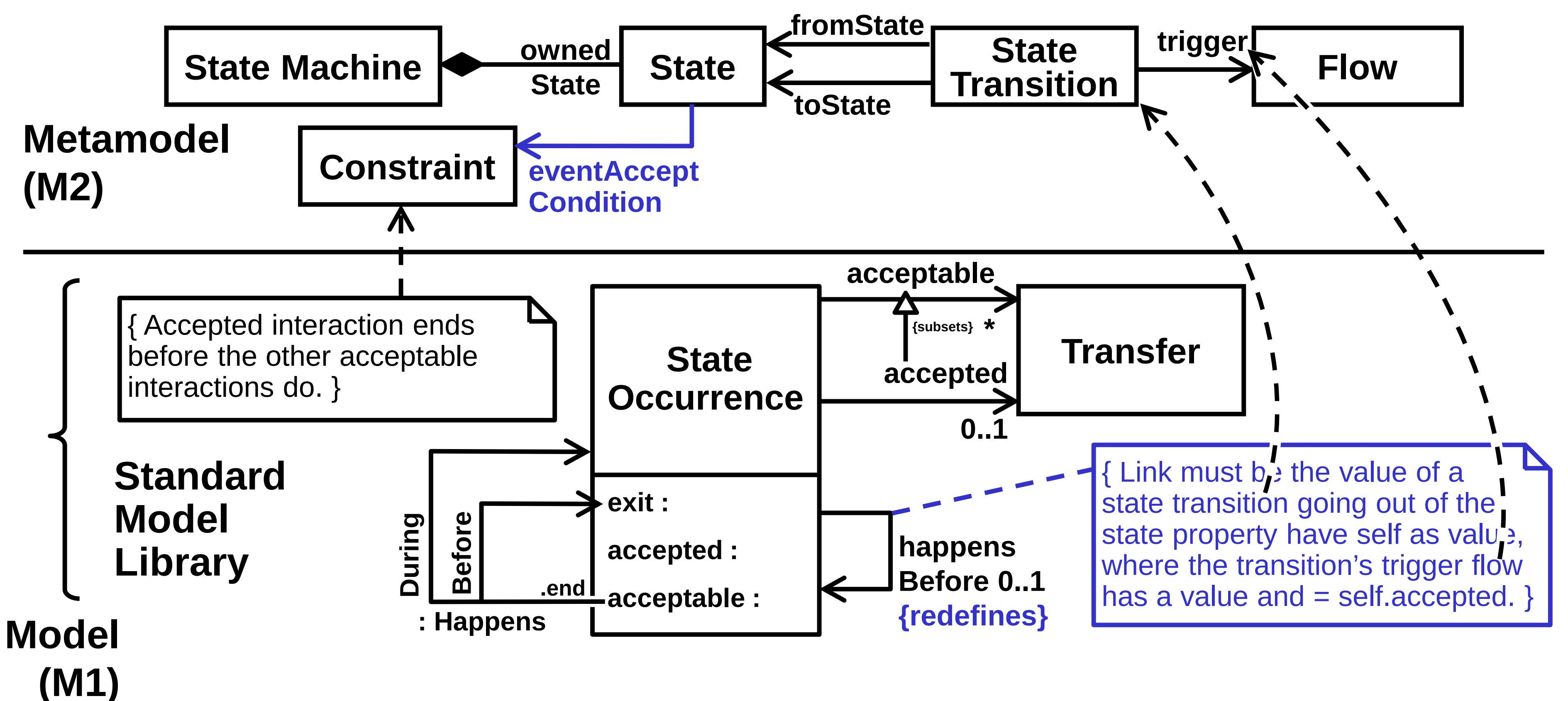
State Machine Problem (P1.2)



Competing Transitions (M1)



Competing Transitions (M1Lib/M2)



- **Library constraints inherited or reused**
  - Acceptable/exit timing moved to library.
  - Transition constraints use M2.
  - Commonly used acceptance constraints.

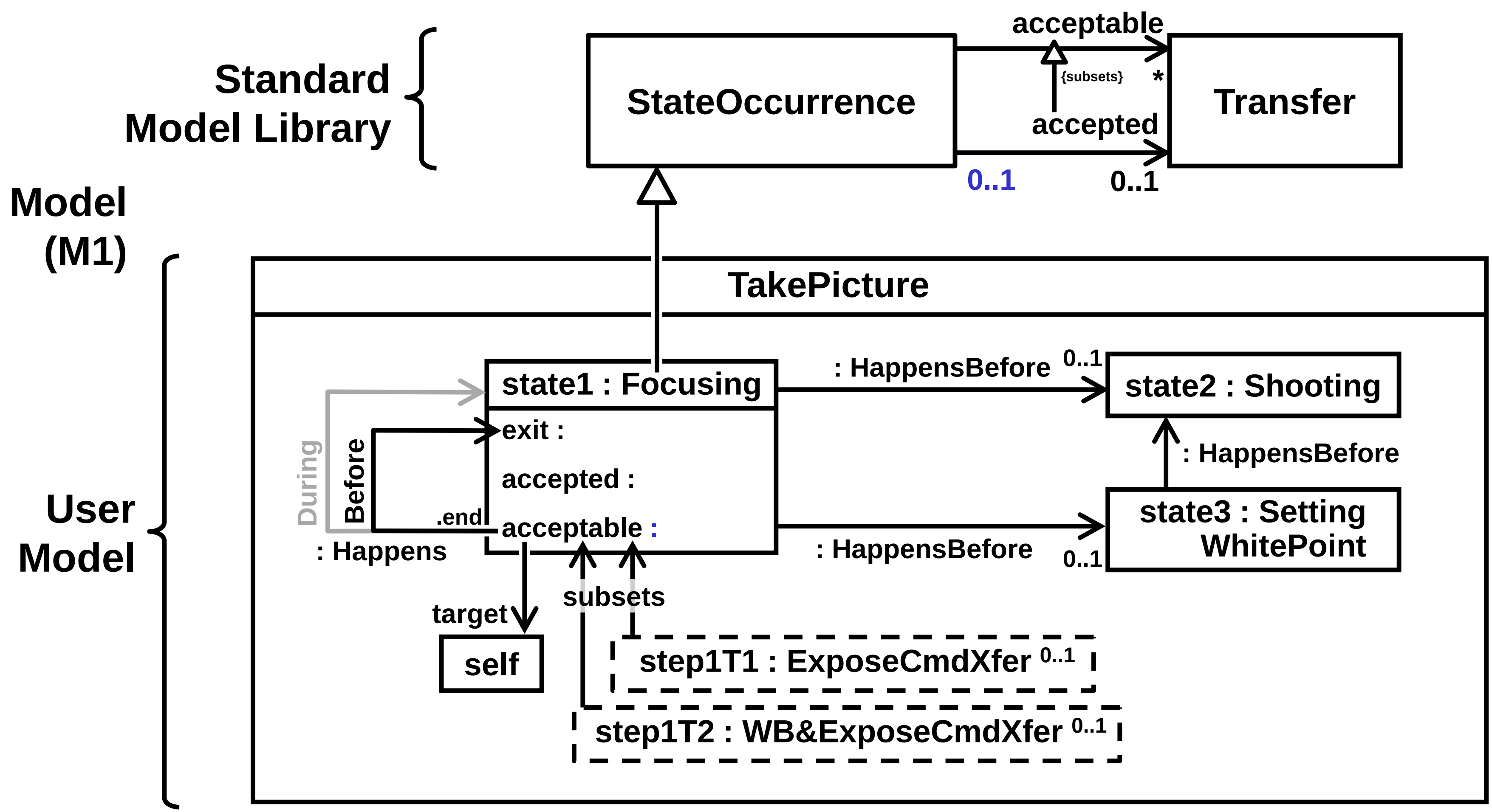
For all models

For models to use as needed

State Machine Solution (P1.3)
(Reacting to past events)

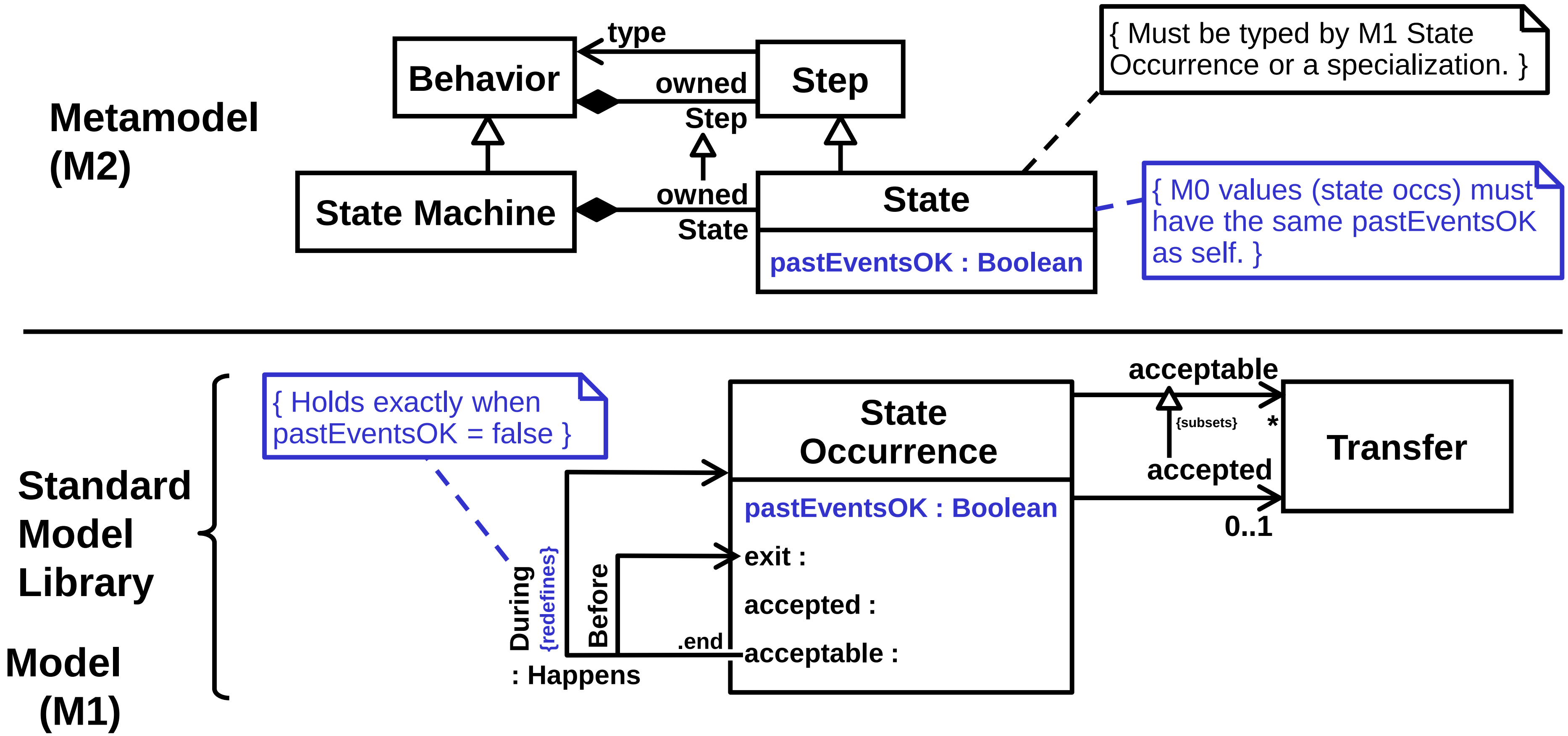
- So far, states are only triggered by **events that arrive during the state.**
- Want to enable states to be triggered by **events that arrive before the state**.
  - **Loosen constraints** against this.

Past Events (M1)



- Events arriving before state are acceptable.
  – But each event can only be accepted once.

Past Events (M1 Library / M2)



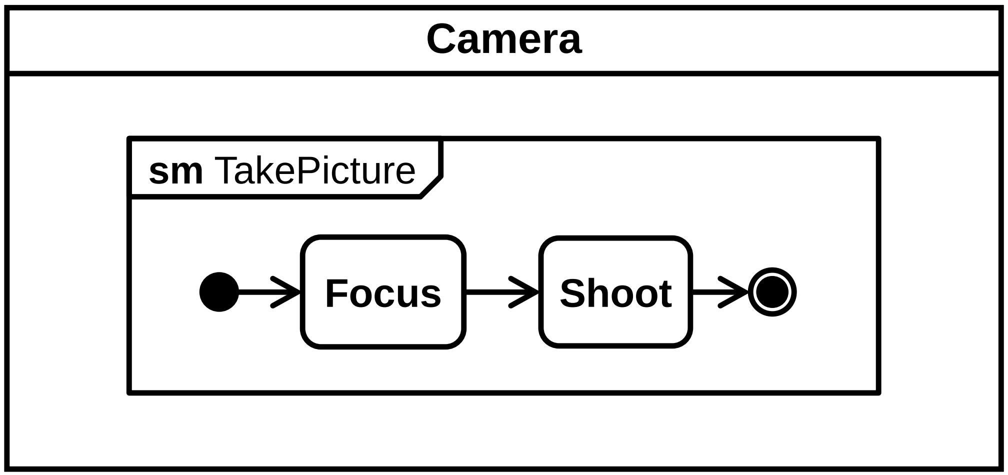
- **HappensDuring redefined to apply as indicated by metamodel boolean.**

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - State machines Part 1, review
  - **State machines Part 2, requirements**
- State Machines Solution, Part 2
  1. Objects reacting to stimuli
  2. Synchronizing state changes
  3. Managing stimuli
- Summary

Behaviors of Objects

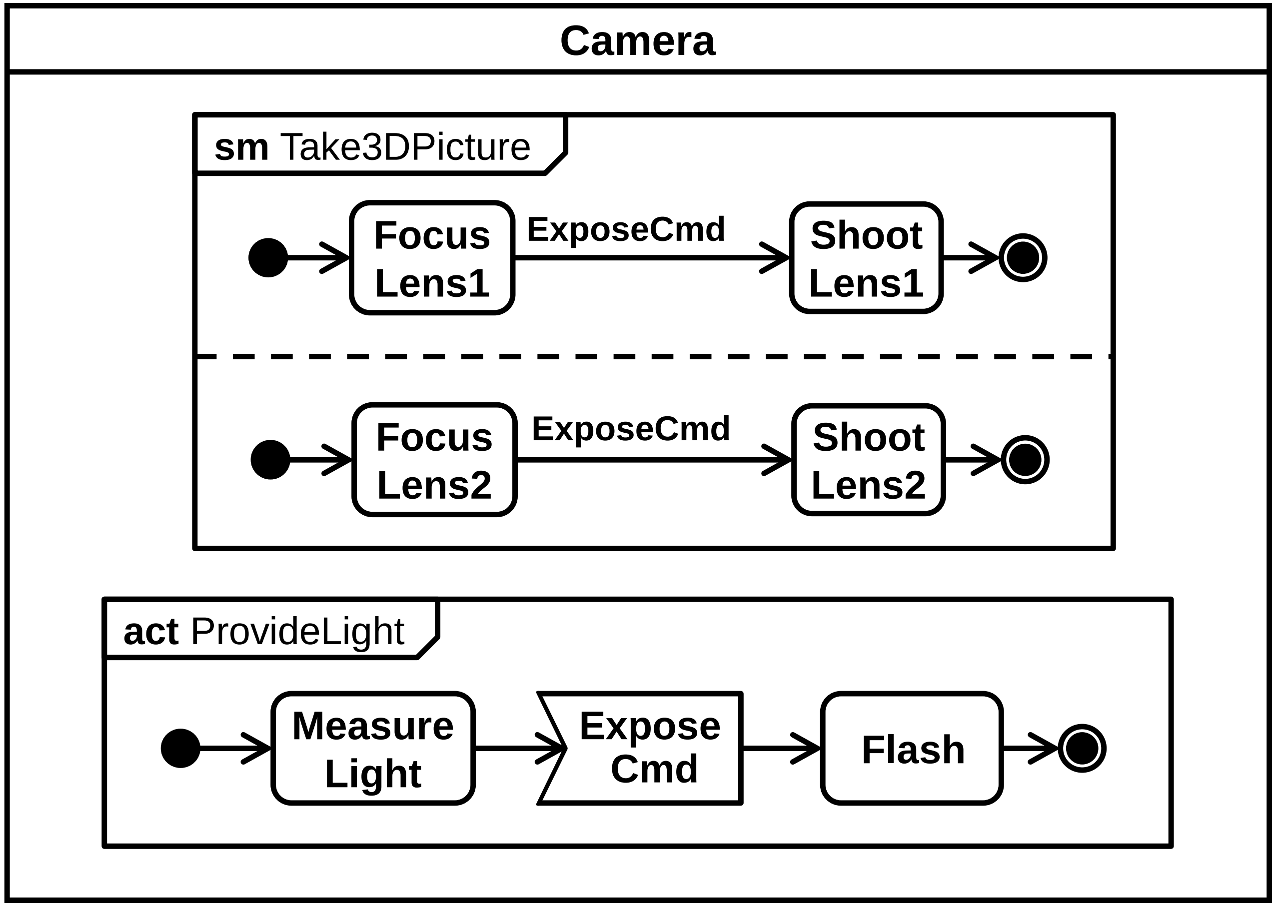
Model (M1)



- **Objects behave.**
  - **UML "classifier behaviors".**
- **States are still states of behaviors …**
- **… reacting to *stimuli arriving at objects*.**
  - **Same as arriving at behavior, except …**

State Machine Problem, Part 2

Model
(M1)



- **Multiple object behaviors can react to the same stimulus** (compare to UML)

State Machine Requirements, P2

1. Must enable objects to react to stimuli.
   - Via behaviors "of" objects.
2. Must synchronize state changes between …
   - Machine regions (part of "run-to-completion").
   - Multiple behaviors for the same object.
3. Must manage multiple stimuli arriving at the same object.

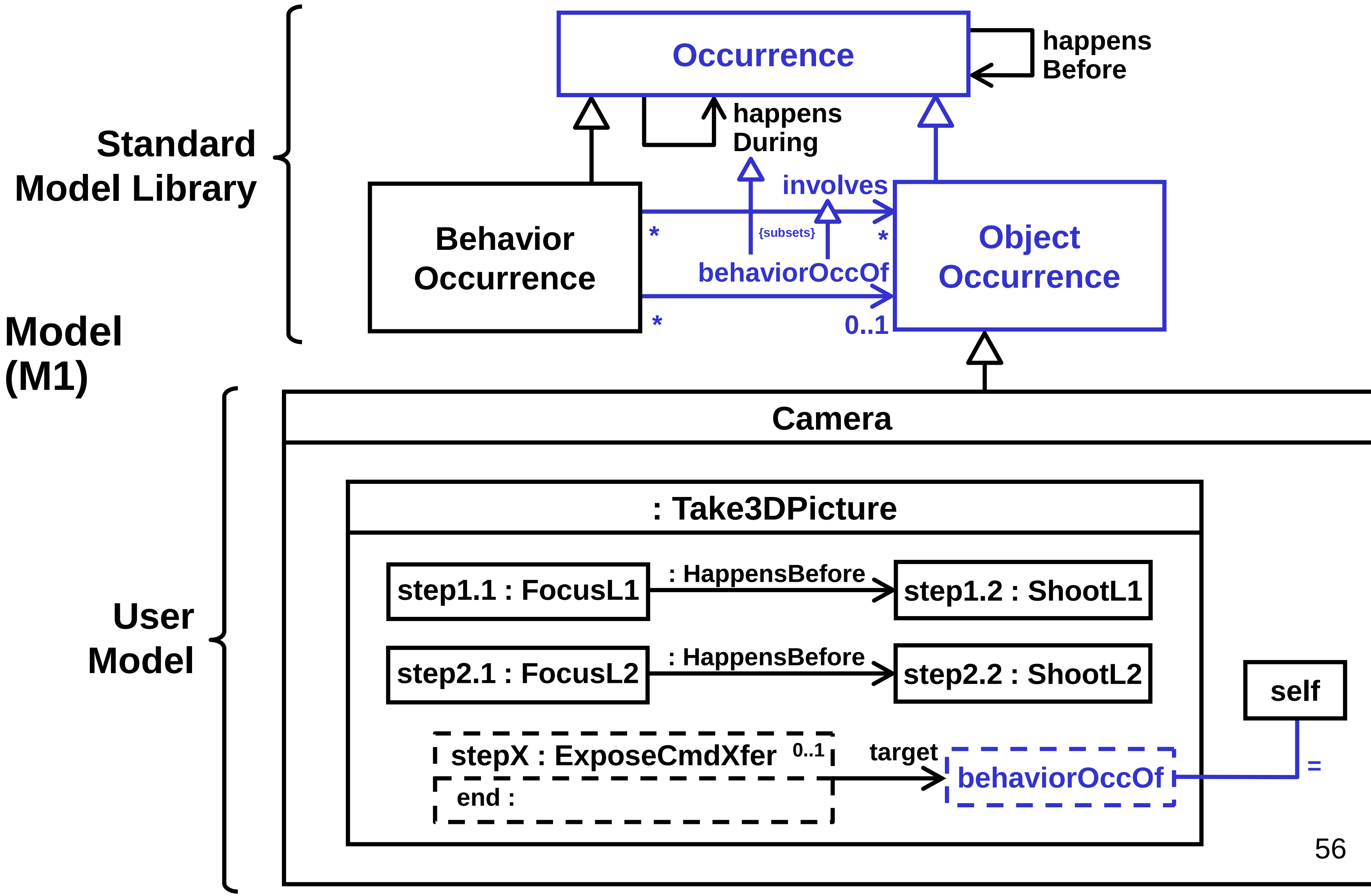
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - State machines Part 1, review
  - State machines Part 2, requirements
- **State Machines Solution, Part 2**
  1. **Objects reacting to stimuli**
  2. Synchronizing state changes
  3. Managing stimuli
- Summary

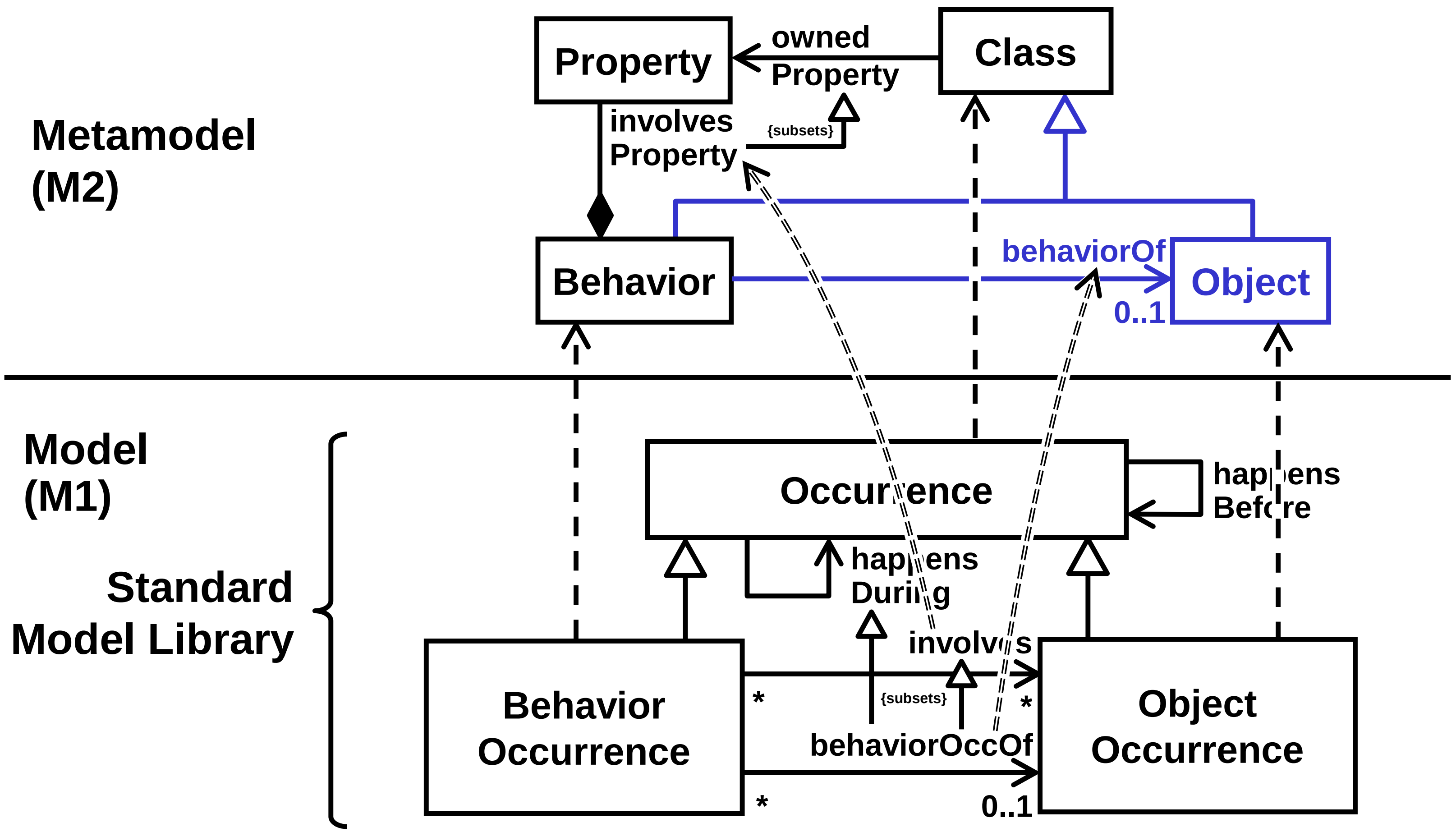
State Machine Solution (Part 2.1)
(Objects reacting to stimuli)

- Objects occur in time also.
  - With different terminology (creation, destruction, etc).
- Behaviors can specify an object to be target of transfers.
  - Assume these are behaviors “of” the object.
  - Multiple behaviors can specify the same object.

Objects Reacting (M1)



Objects Reacting (M2)

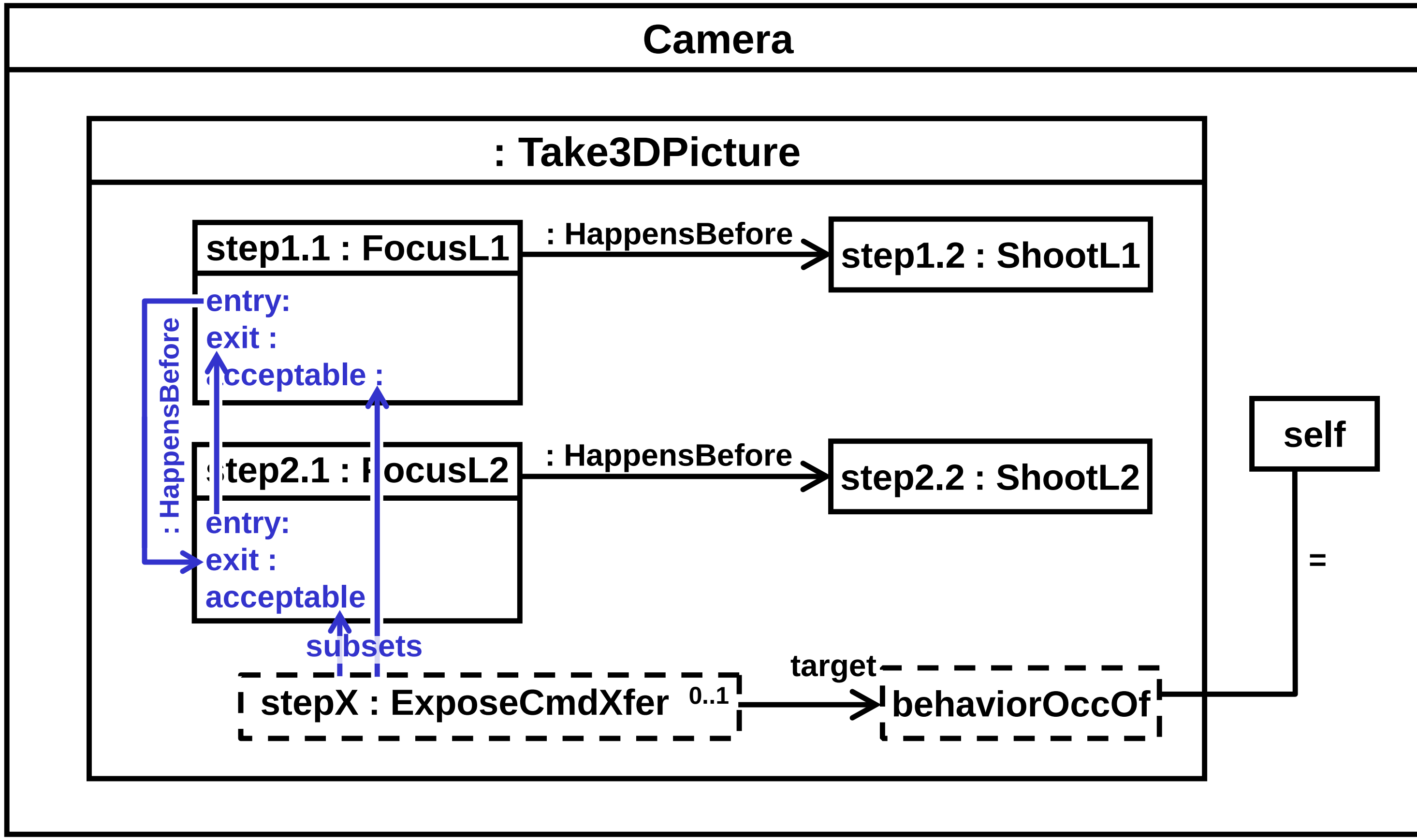


Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - State machines Part 1, review
  - State machines Part 2, requirements
- State Machines Solution, Part 2
  1. Objects reacting to stimuli
  2. **Synchronizing state changes**
  3. Managing stimuli
- Summary

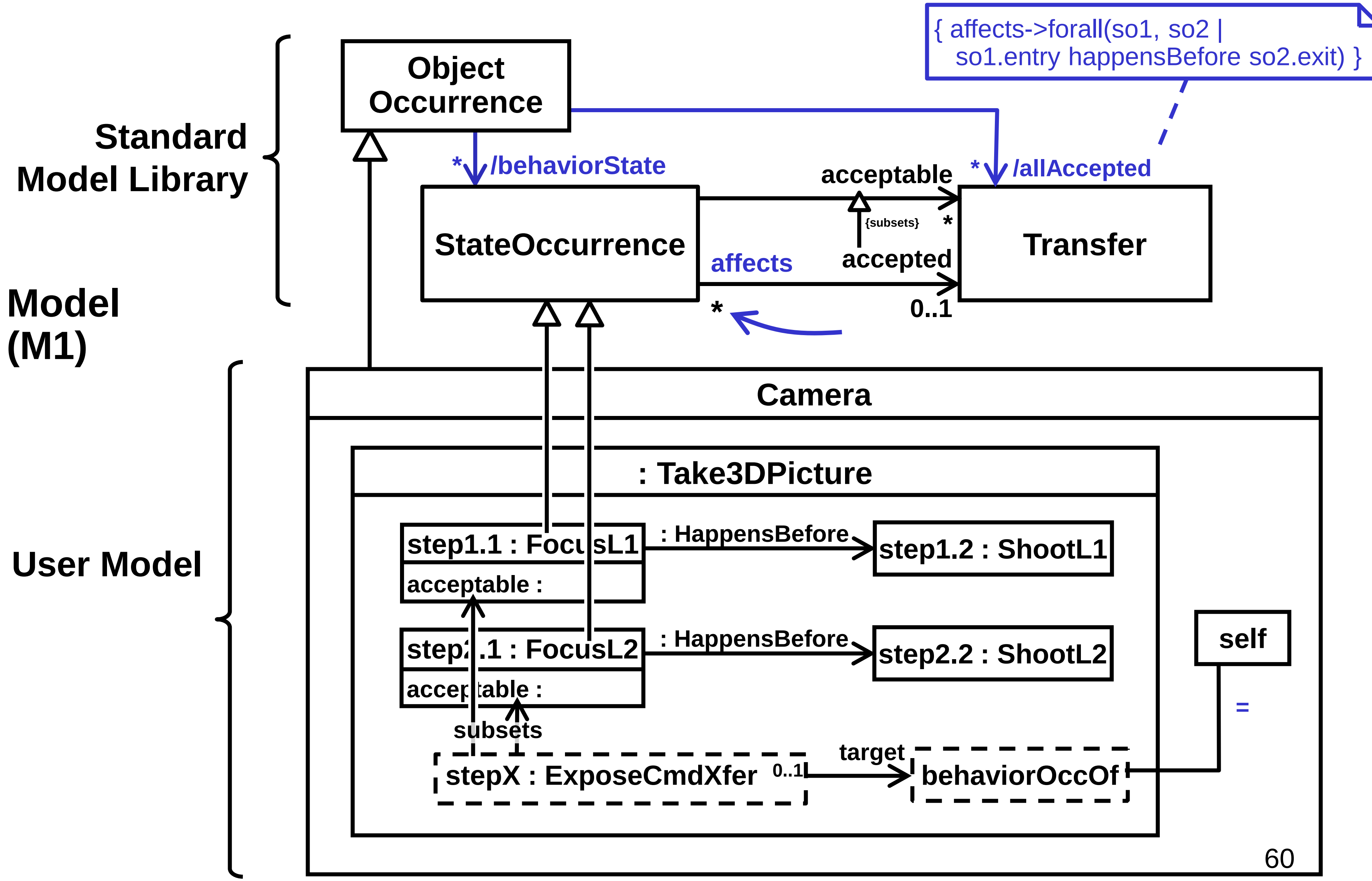
Synchronized Regions (M1)

Model
(M1)



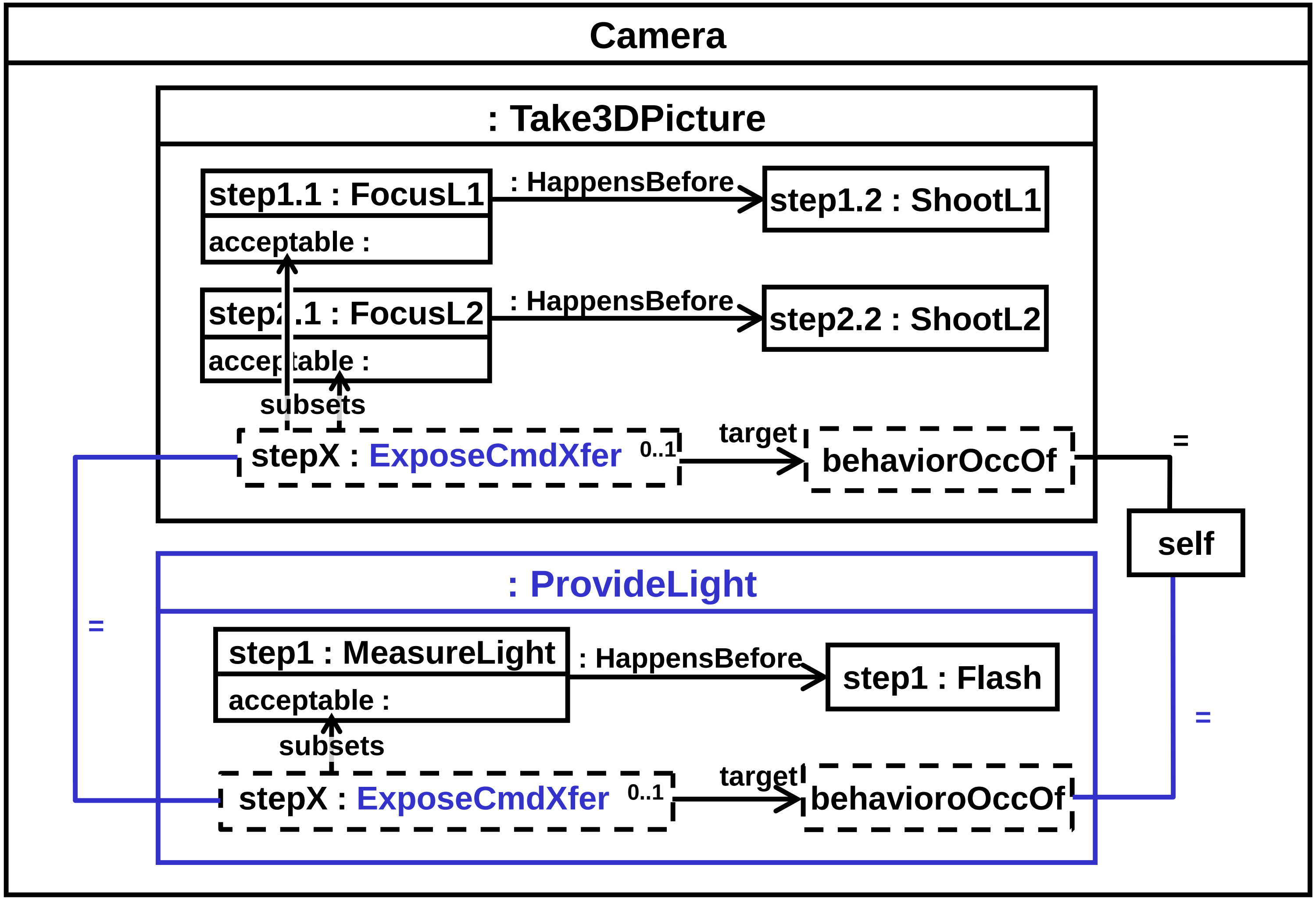
- Transitions triggered by same event.
  – Entries happen before exits across states also.
  – Other timing not restricted in UML.

Synchronized Regions (M1, Lib)



Objects, Multiple Behaviors (M1)

Model (M1)



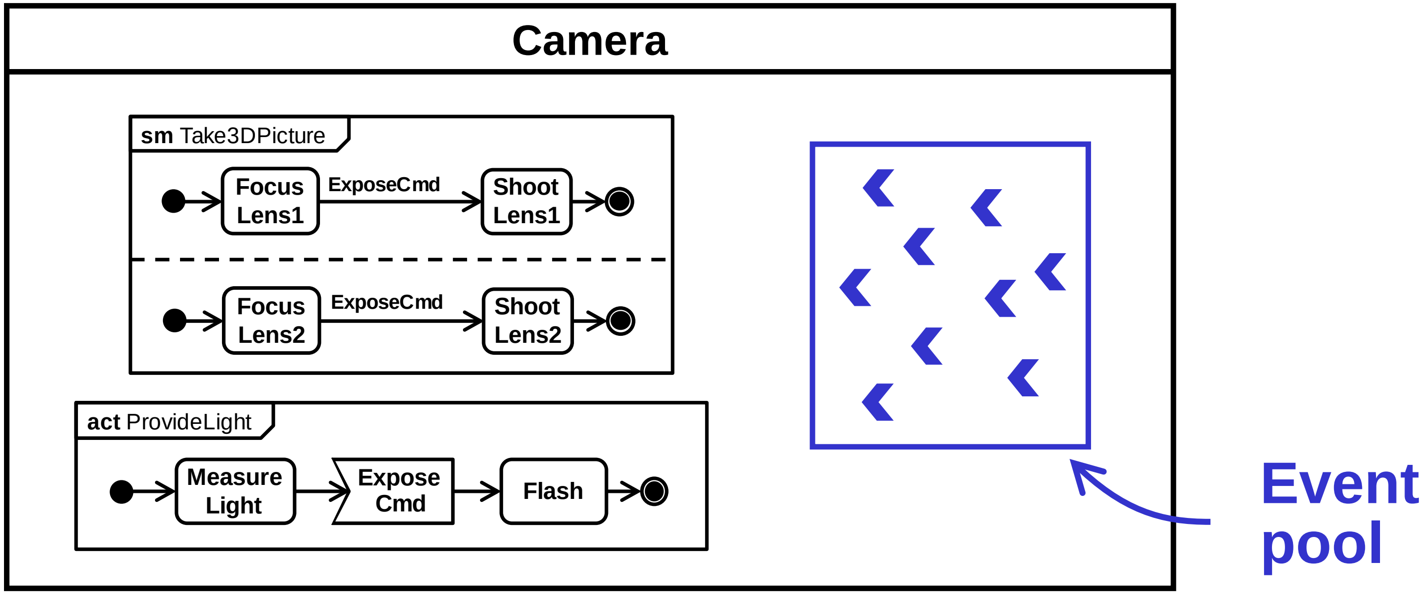
- **Multiple behaviors reacting to same event.**
  - **Treated as regions (compare to UML).**

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - State machines Part 1, review
  - State machines Part 2, requirements
- State Machines Solution, Part 2
  1. Objects reacting to stimuli
  2. Synchronizing state changes
  3. **Managing stimuli**
- Summary

UML Event Handling

Model
(M1)

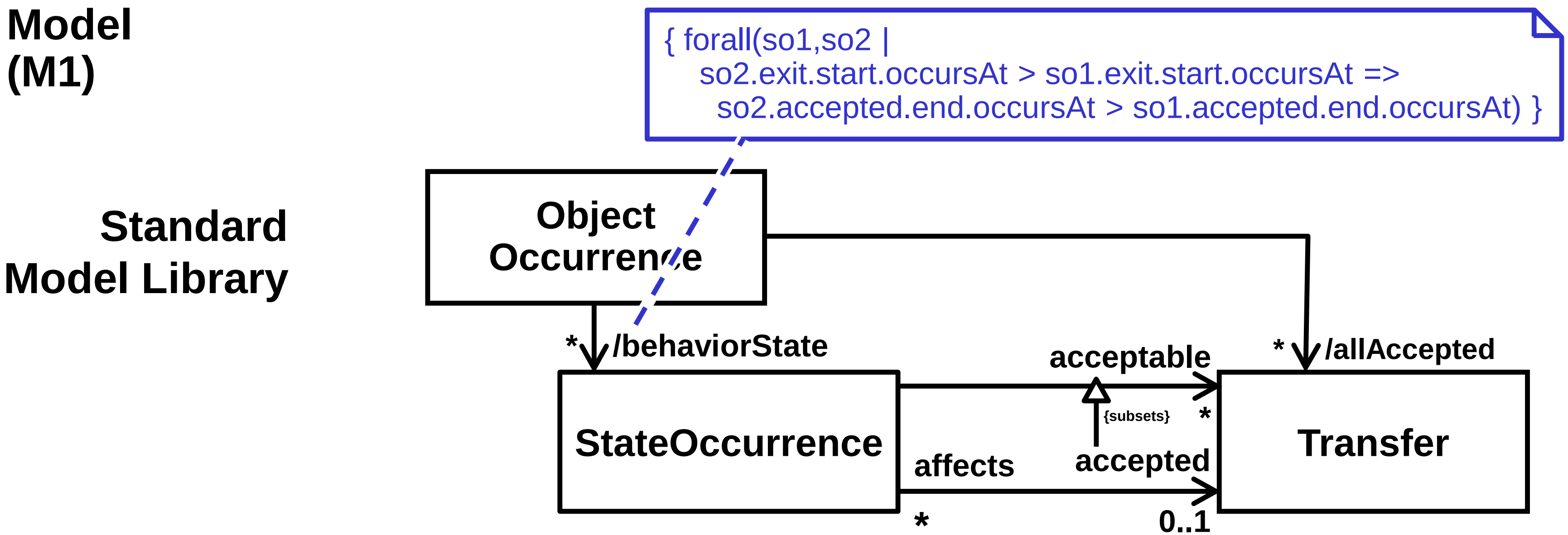


- **Objects have a single “pool” of events.**
- **One event at a time is**
  - **Checked against transition triggers …**
    - **Matching transitions are taken.**
  - **Removed from the pool …**
    - **Whether or not it triggers transitions (exception later)**

Onto-izing UML Event Handling

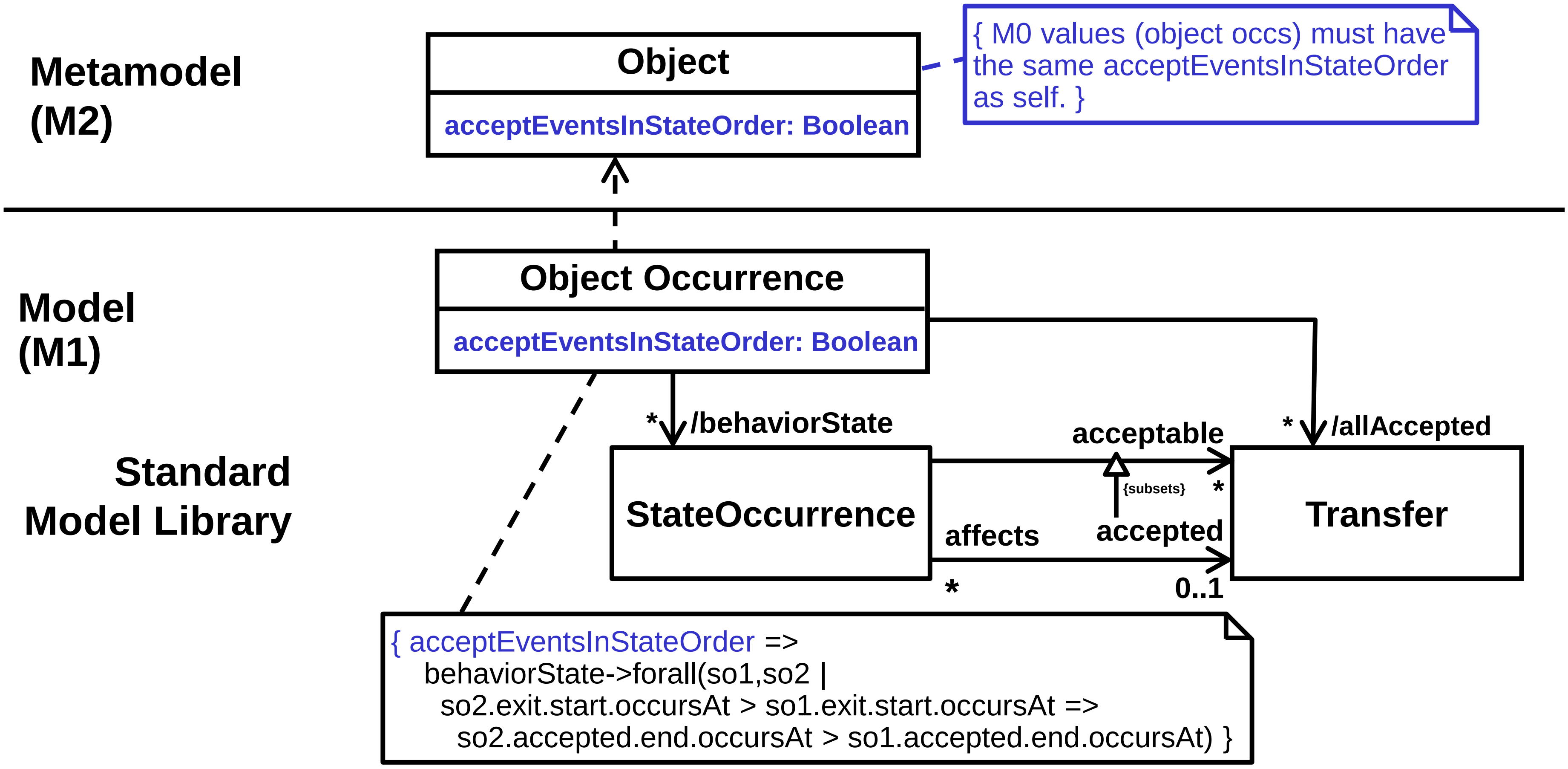
- UML describes an event handling procedure (execution engine).
- Classification modeling
  - Gives conditions required of valid executions.
    - No procedure for handling events.
  - Triggers satisfy (match) these conditions, or not.
    - Events are not removed from a pool.
  - More complex event handling procedures = more complex classification conditions.

Pool as Queue (M1)



- UML: Check events in the order they arrived.
- Accepted events "arrive" in same order as transitions out of state occs that "accept" them.
  - Ends of transfers to self identified as accepted by state occurrences must be in the same time order as those occurrences are left.

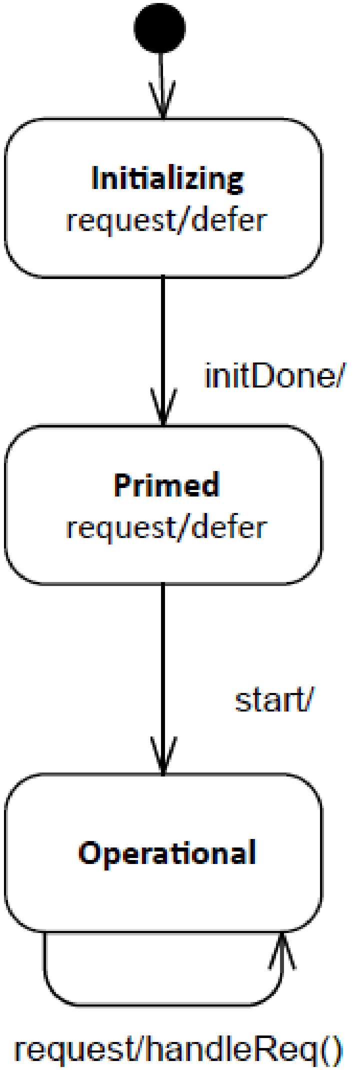
Pool as Queue (M2)



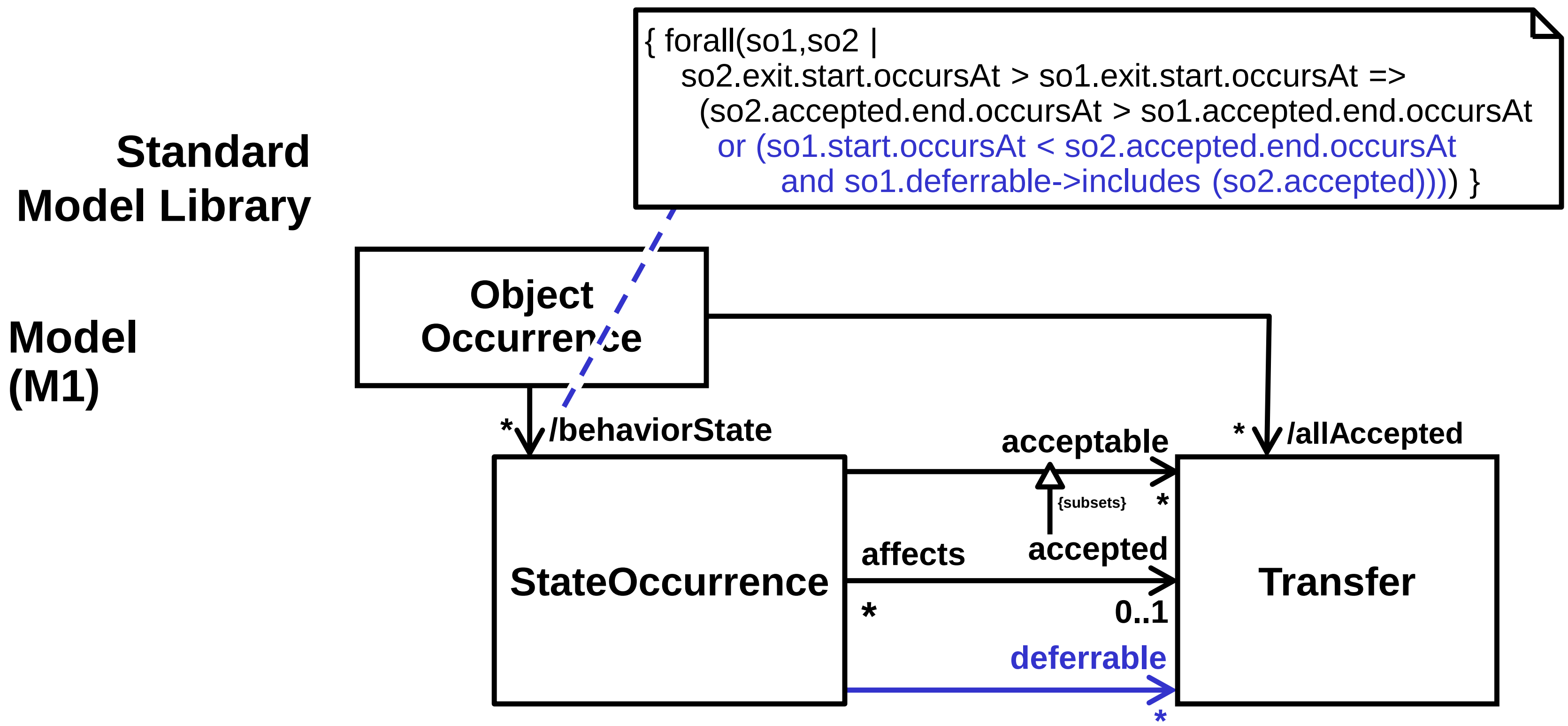
- **Constraint applies as indicated by metamodel boolean.**

Deferrable Events

- UML states can indicate some events remain in the pool.
  - Even though they were checked against transitions triggers.
- Transfers to objects that do not violate classification conditions.
  - Deferral specified as exceptions to normal conditions.

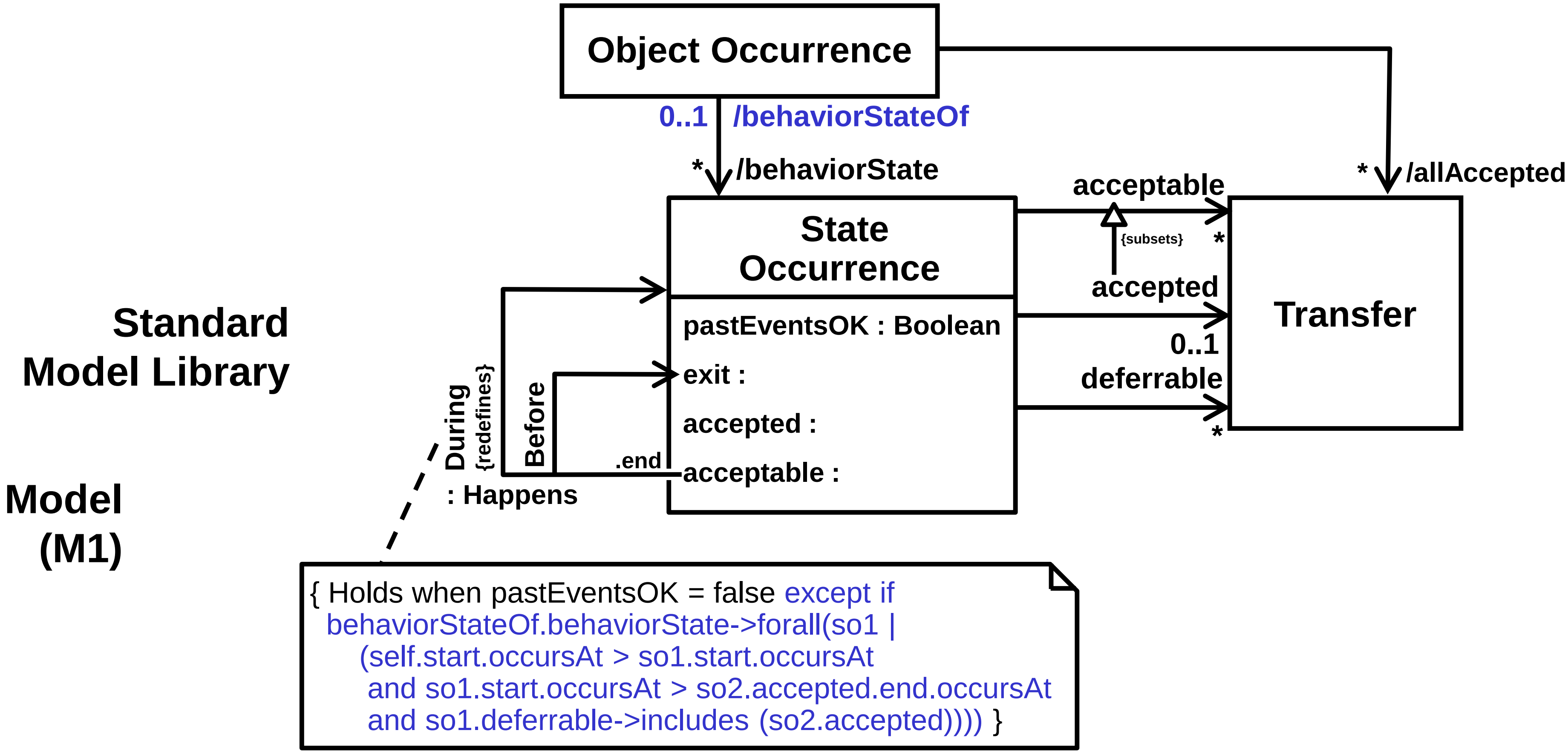


Deferrable Events (in Queuing)



- **Transfer arrival condition loosened for deferrable events.**

Deferrable Events (in Past Events)



- **Past event condition loosened for deferrable events.**

State Machine TBD (Post P1)

- Events arriving at objects.
- Concurrent regions.
- More complex event handling.
  - Deferrable events
  - Completion events
- Event content
- Transitions
  - Guards
  - Internal
- Pseudostates
  - State entry / exit points
  - History, etc.
- Submachine states

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - State machines Part 1, review
  - State machines Part 2, requirements
- State Machines Solution, Part 2
  1. Objects reacting to stimuli
  2. Synchronizing state changes
  3. Managing stimuli
- **Summary**

Summary

- **Objects react to stimuli via**
  - **Transfers targeting objects.**
  - **Behaviors reacting to these transfers arriving.**
    - **For any kind of behavior that reacts to events.**
- **State changes synchronized by**
  - **Constraining exit behavior timing across regions and behaviors.**
- **Stimuli managed by timing constraints on events and state occurrences.**
  - **Same effect as UML event processing (mostly).**
- **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**
- **Object-orientation as Composite Structure**
  - **http://doc.omg.org/ad/18-09-07**
- **State Machines as Composite Structure, Part 1**
  - **http://doc.omg.org/ad/18-12-09**
- **Earlier slides (more 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**