(Onto) Logical Requirements and Designs

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

Raphael Barbau
Engisis

National Institute of Standards and Technology

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Requirements and designs, requirements
- Requirements and designs, Solution
  1. Satisfaction, derivation, and refinement
  2. Operation and effect
- Summary

Overview

- RoadMap
- Motivation
  - Behavior, review
  - Requirements and designs, requirements
- Requirements and designs, Solution
  1. Satisfaction, derivation, and refinement
  2. Operation and effect
- Summary

Behavior as Composite Structure

Onto Requirements and Designs
(this one)

Onto 4D
(ad/19-09-07)

Onto Activities
(ad/19-06-02)

Onto State Machines, Parts 1 & 2
(ad/18-12-09, 19-03-02)

Onto Interactions (ad/18-06-11)	**Onto OO** (ad/18-09-07)

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

Overview

- RoadMap
- Motivation
  – Behavior, review
  – Requirements and designs, requirements
- Requirements and designs, Solution
  1. Satisfaction, derivation, and refinement
  2. Operation and effect
- Summary

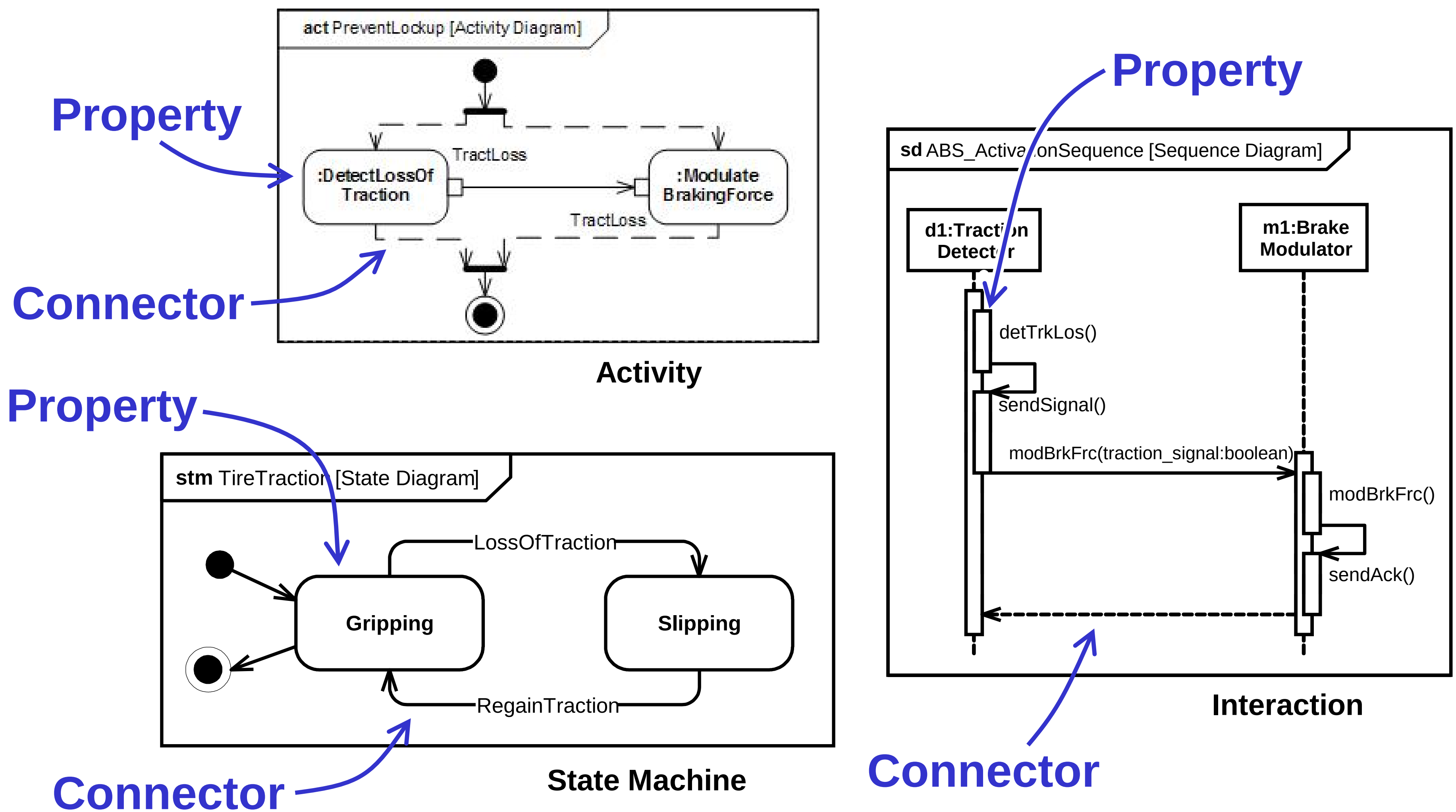
Original Problem

- 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

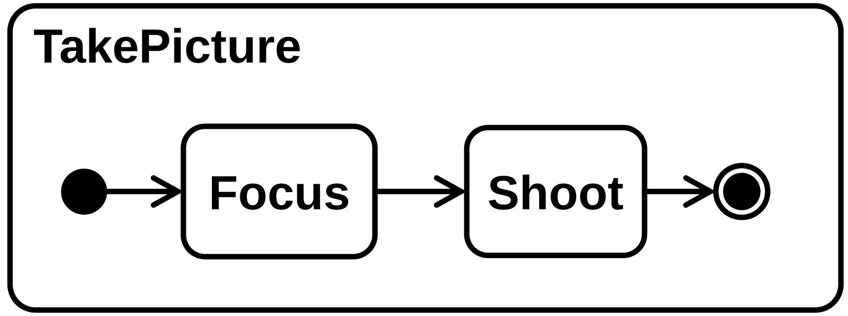
- Treat behaviors as assemblies of other behaviors.
  - Like objects are assemblies of other objects.
- Assembly = UML internal structure
  - Pieces represented by properties.
  - Put together by connectors.
- Put all behavior diagrams on the same underlying behavior assembly model.

Behaviors as Composite Structure

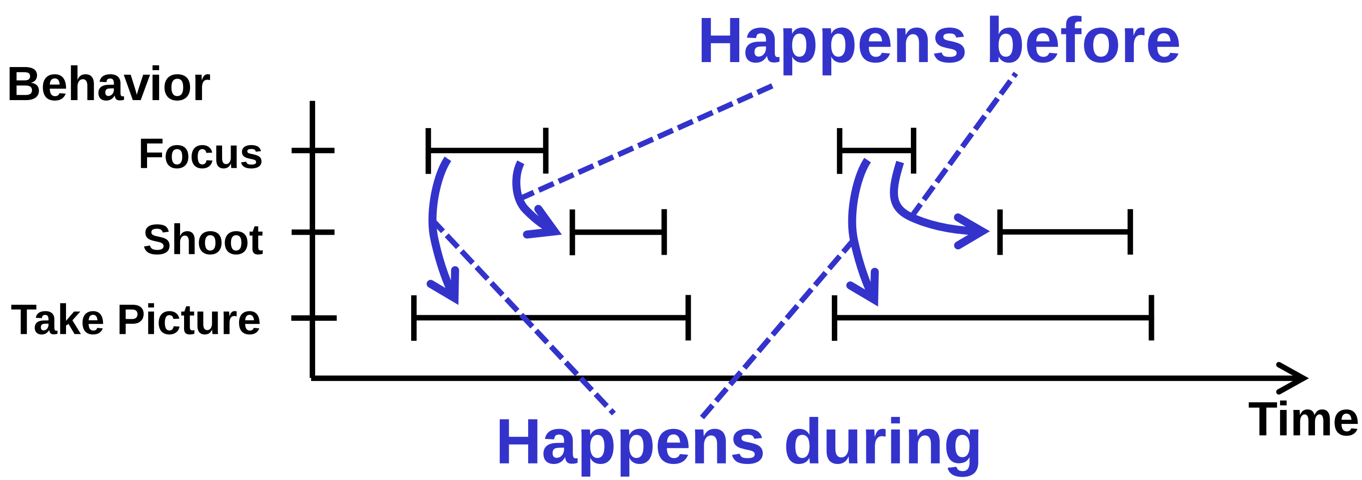


Behavior as Timing Constraints

Model (M1)



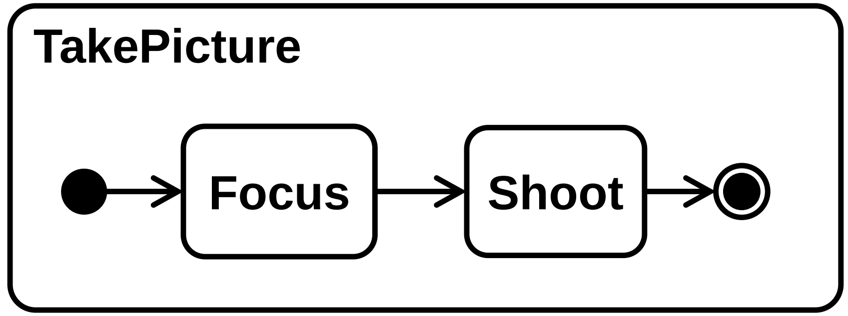
Things Being Modeled (M0)



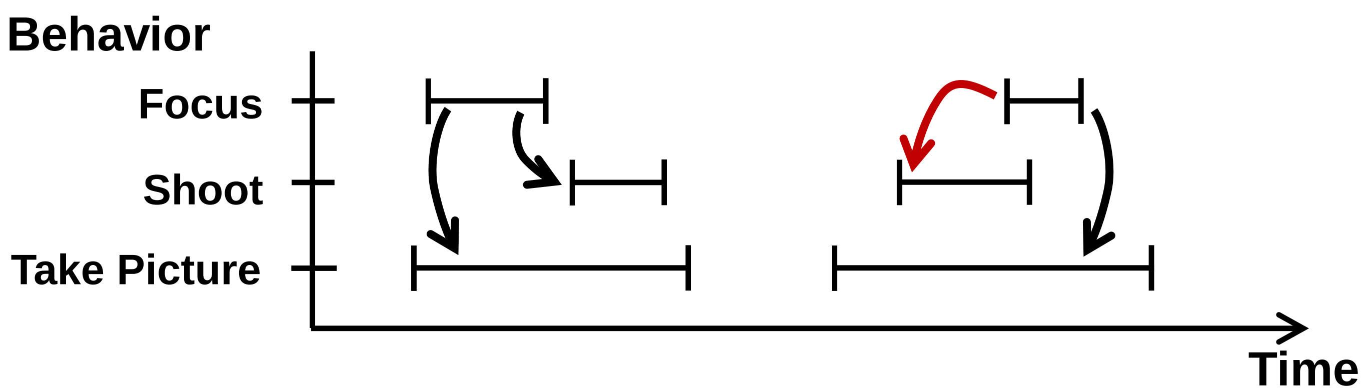
- **Behaviors model “things” happening over time.**
  - **With temporal relations (time constraints) between them.**

Behavior as Timing Constraints

Model (M1)

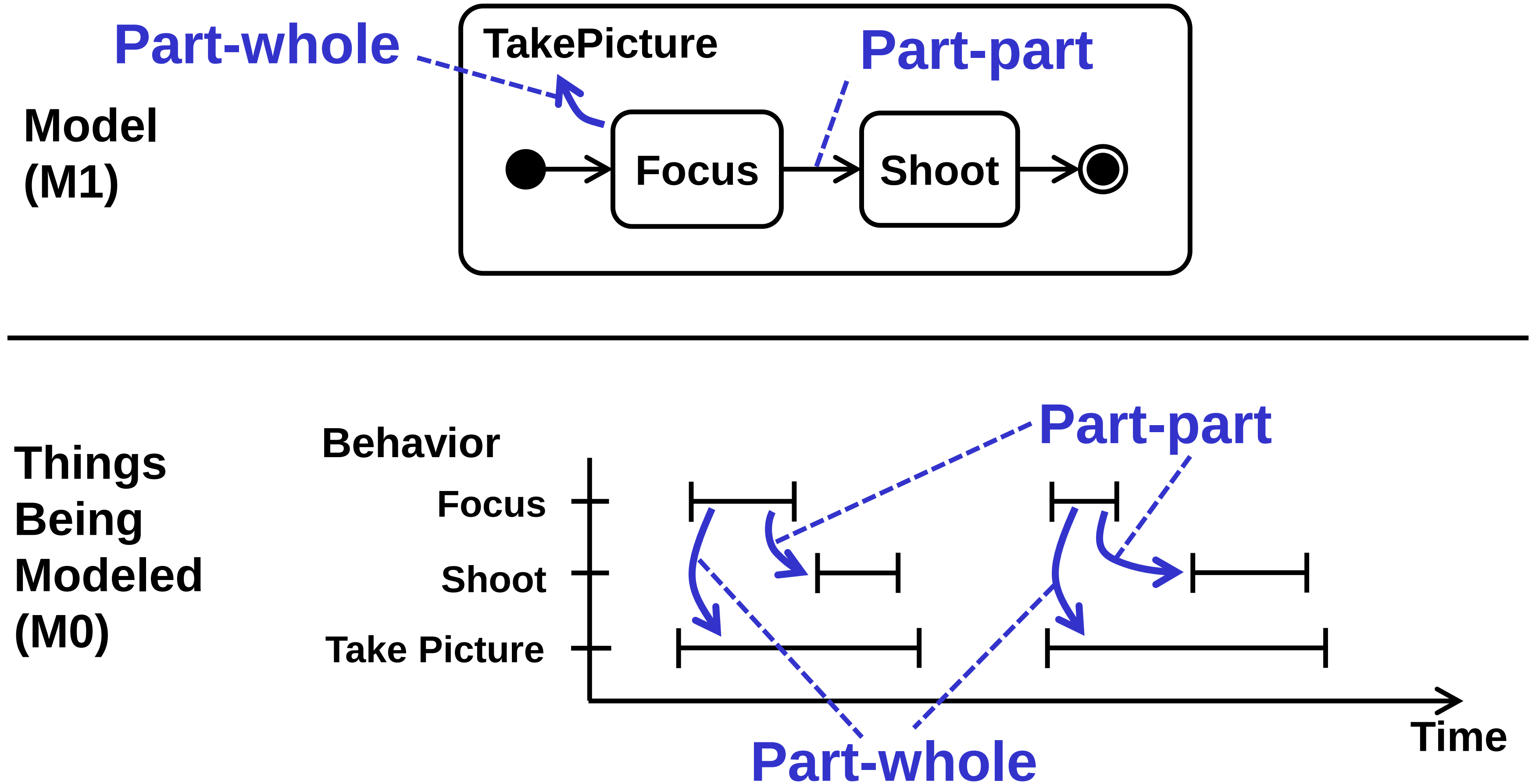


Things Being Modeled (M0)



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

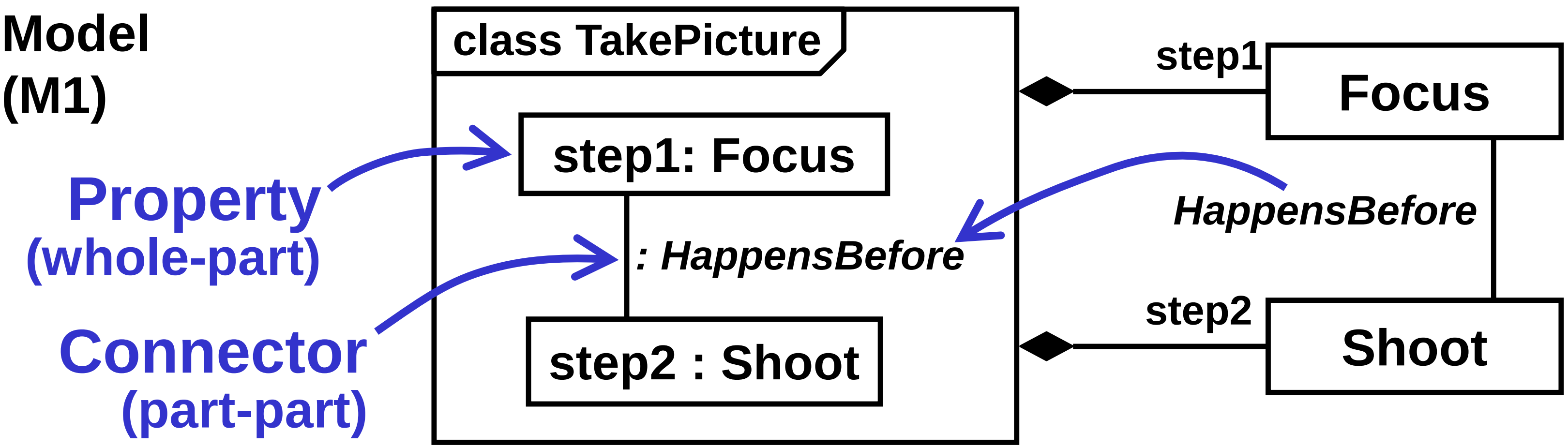
Behavior as “Composite Timing”



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

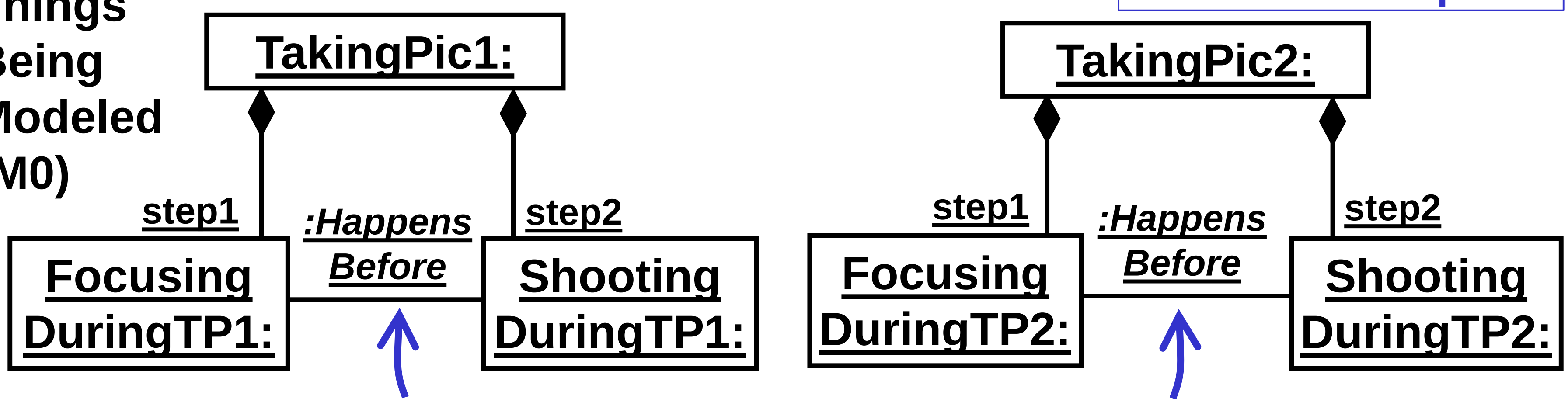
Behavior as "Composite Timing"

Model (M1)



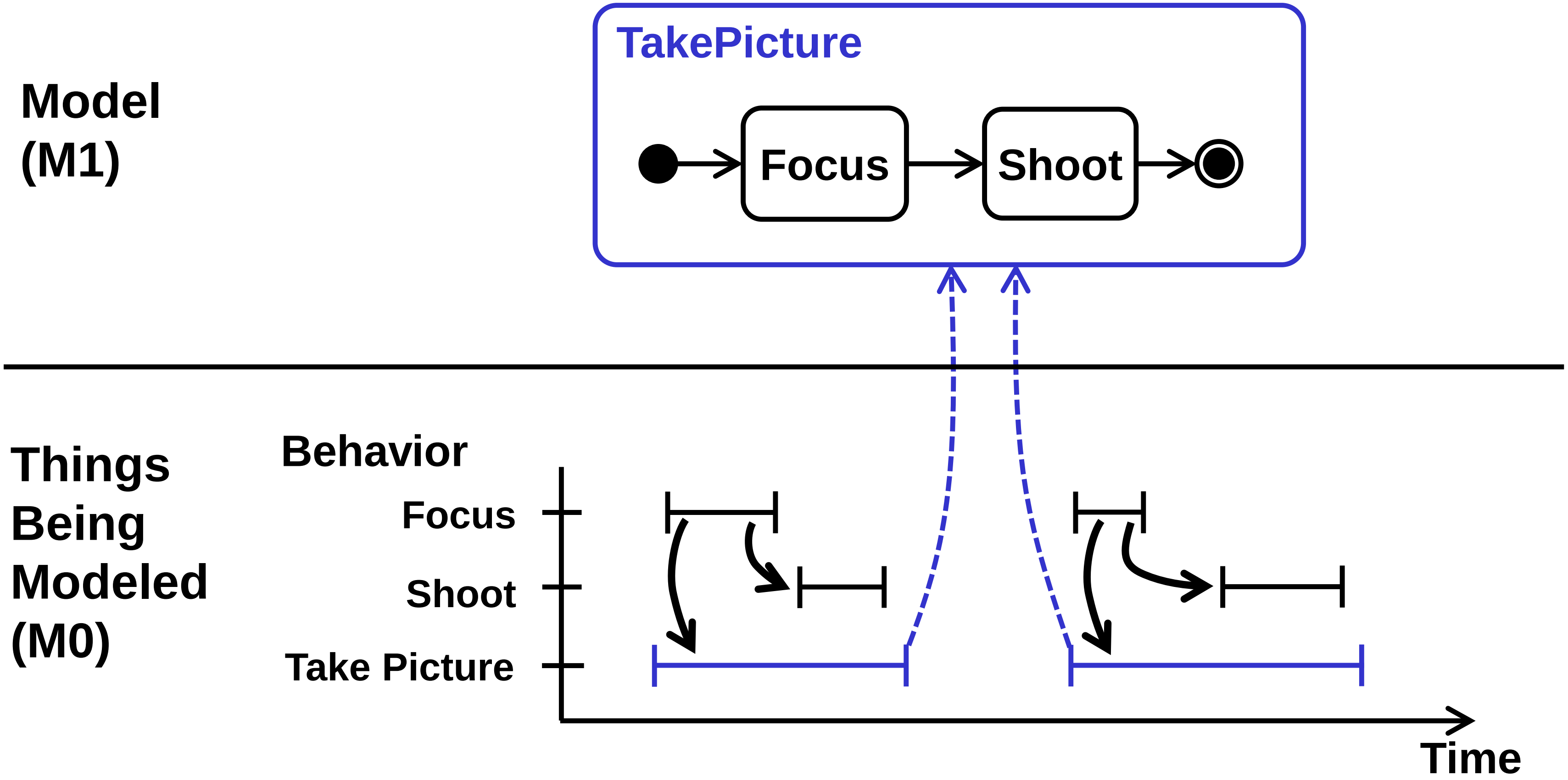
Things Being Modeled (M0)

Not instance specs



Focusing before shooting in same taking picture

Model and Things Being Modeled



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

M0 → M1 Synonyms

Classified by

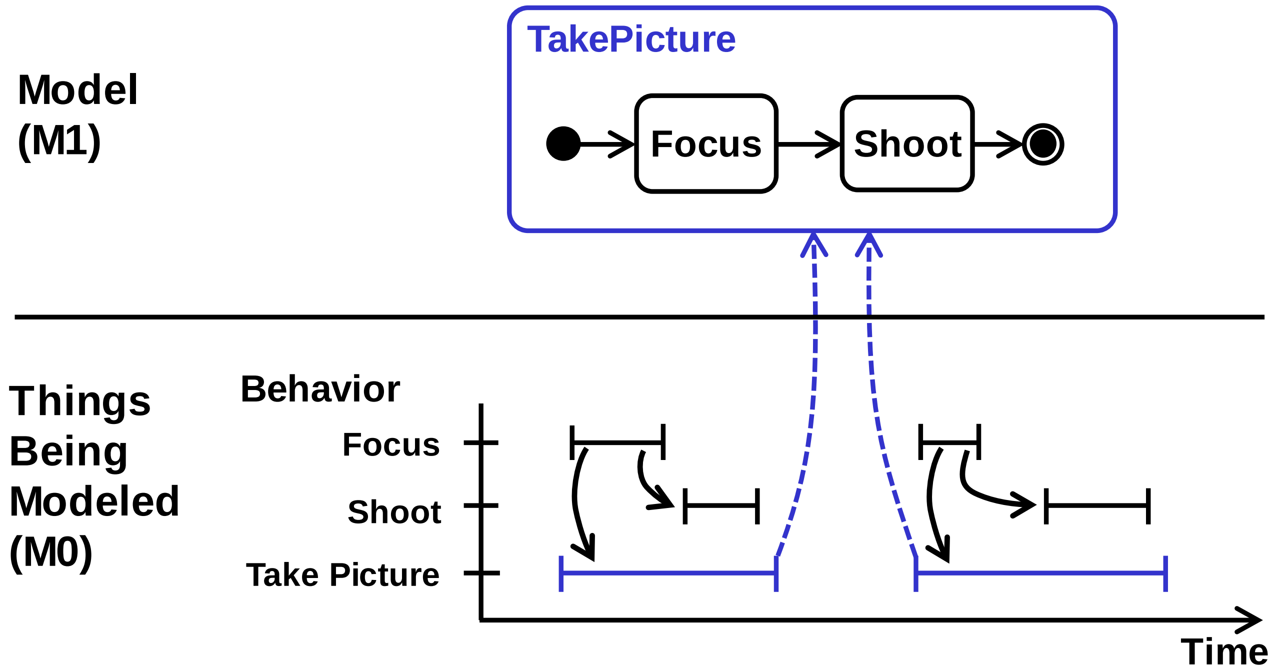
Modeled by

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.

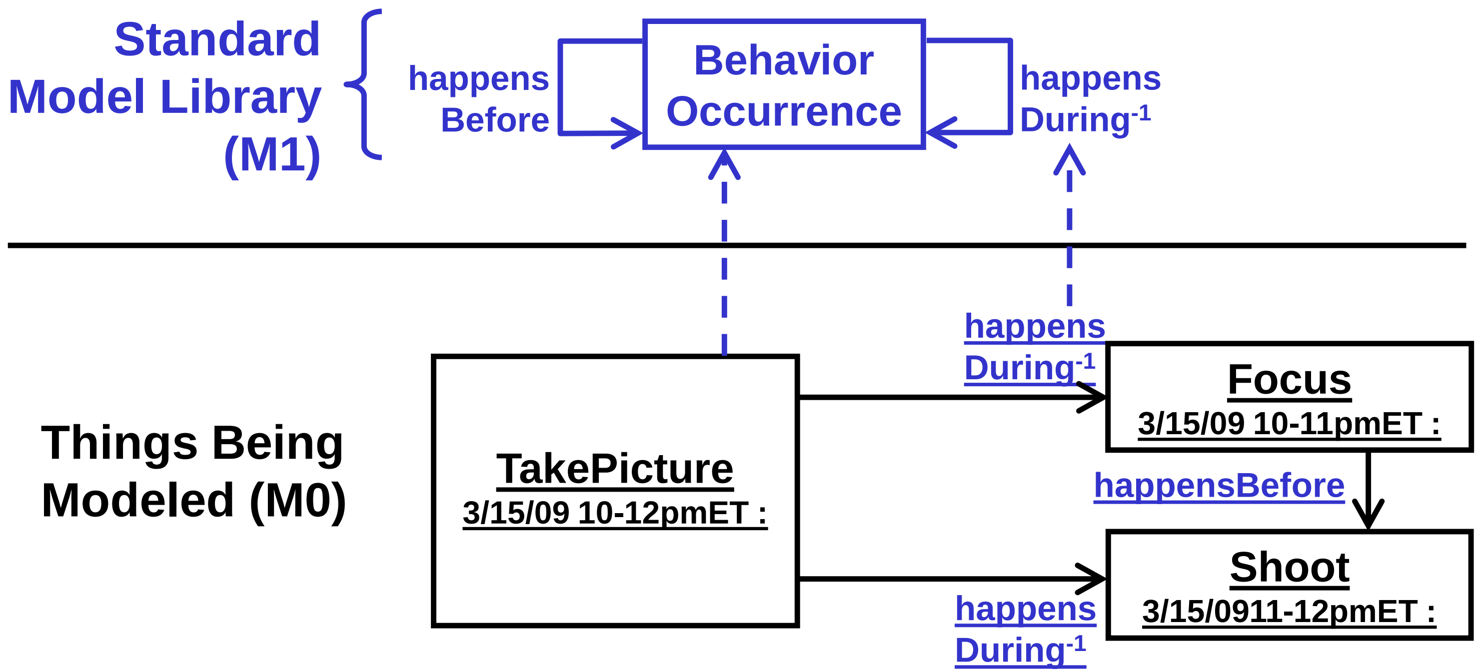
Focus
3/15/09 10-11pmET :

TakePicture
3/15/09 10-12pmET :

Shoot
3/15/0911-12pmET :

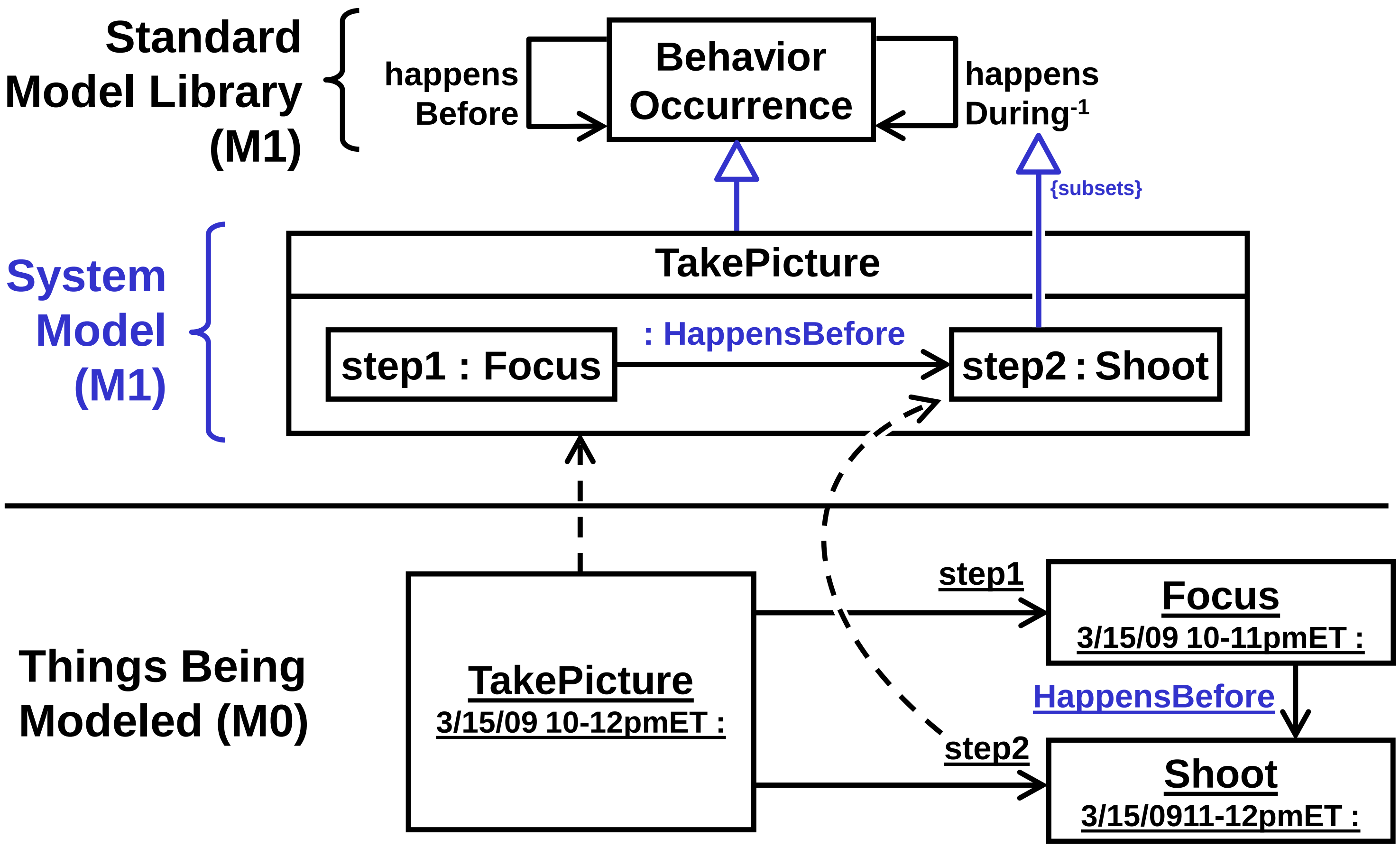
- "Things" that occur in time
  - Eg, taking a picture, focusing, etc.
  - Not "behaviors", "actions", etc.

Behavior: What's in Common?



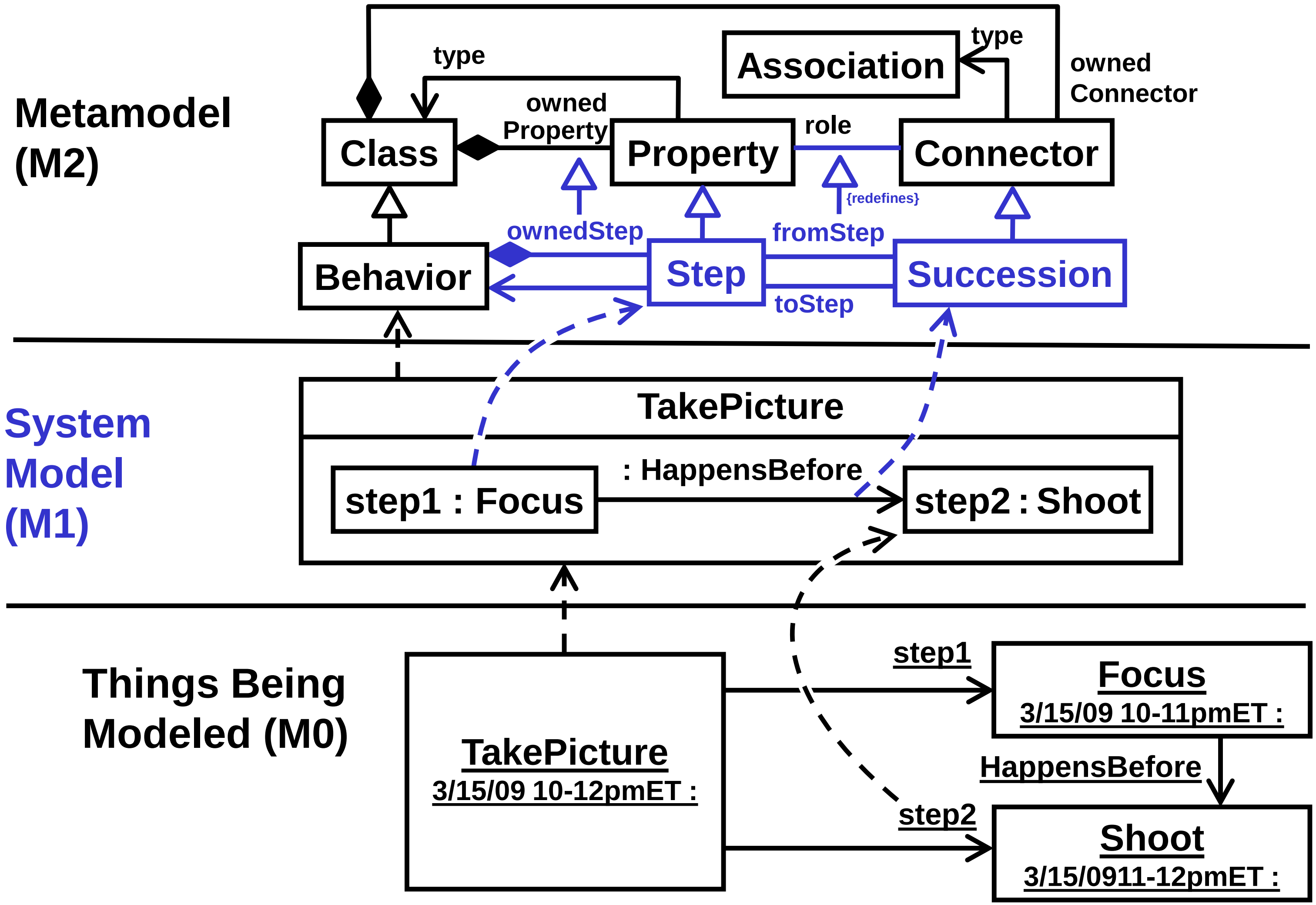
- They happen before or during each other.
  - Construct M1 library for this.
  - Use it to classify things being modeled.

Behavior: Use Library



- Specialize library classes and subset/redefine library properties.

Behavior: Too repetitive at M1?

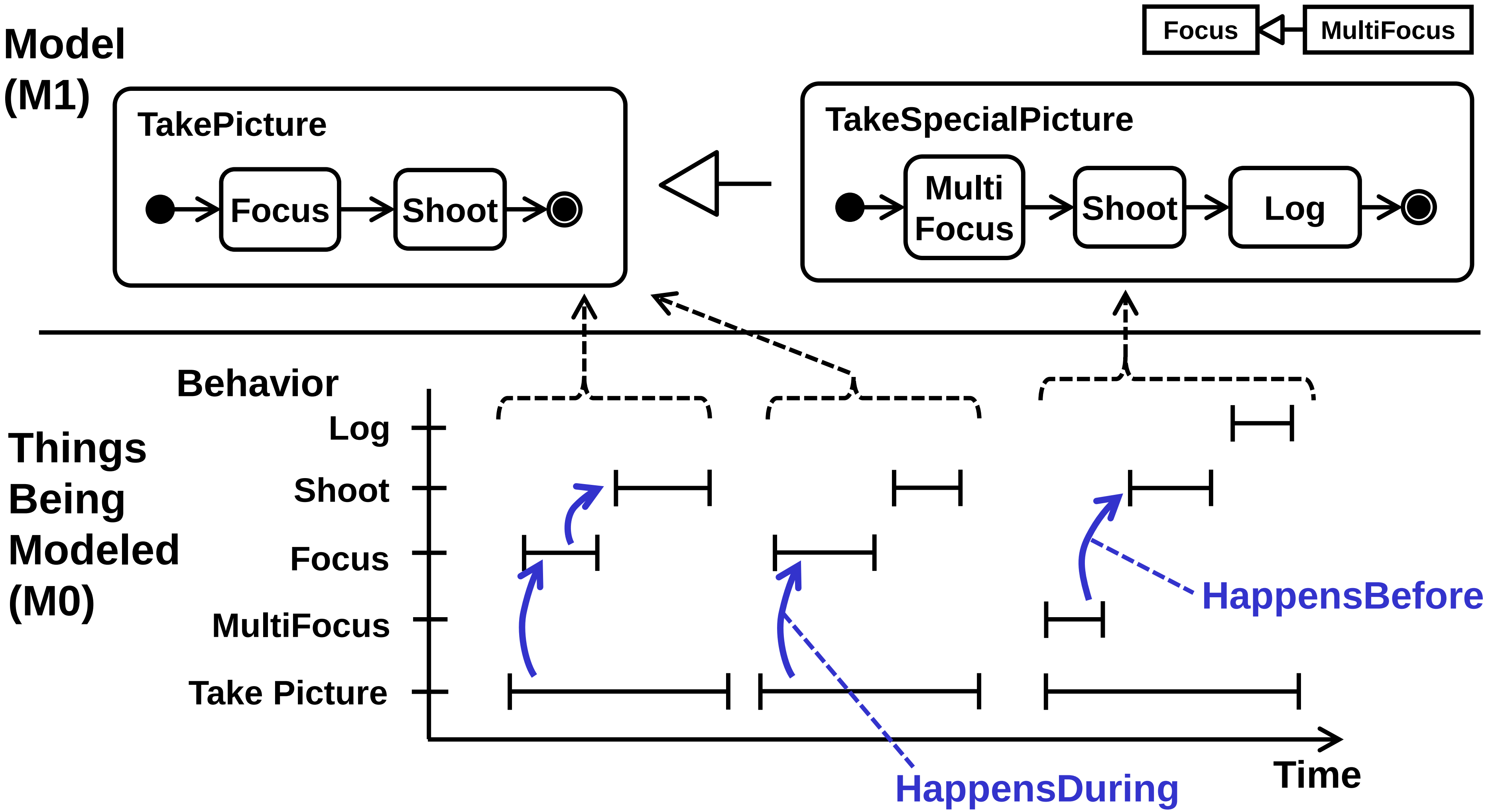


- Capture M1 patterns in M2 elements.
  - Tools apply patterns automatically.

Benefits: Original Problem

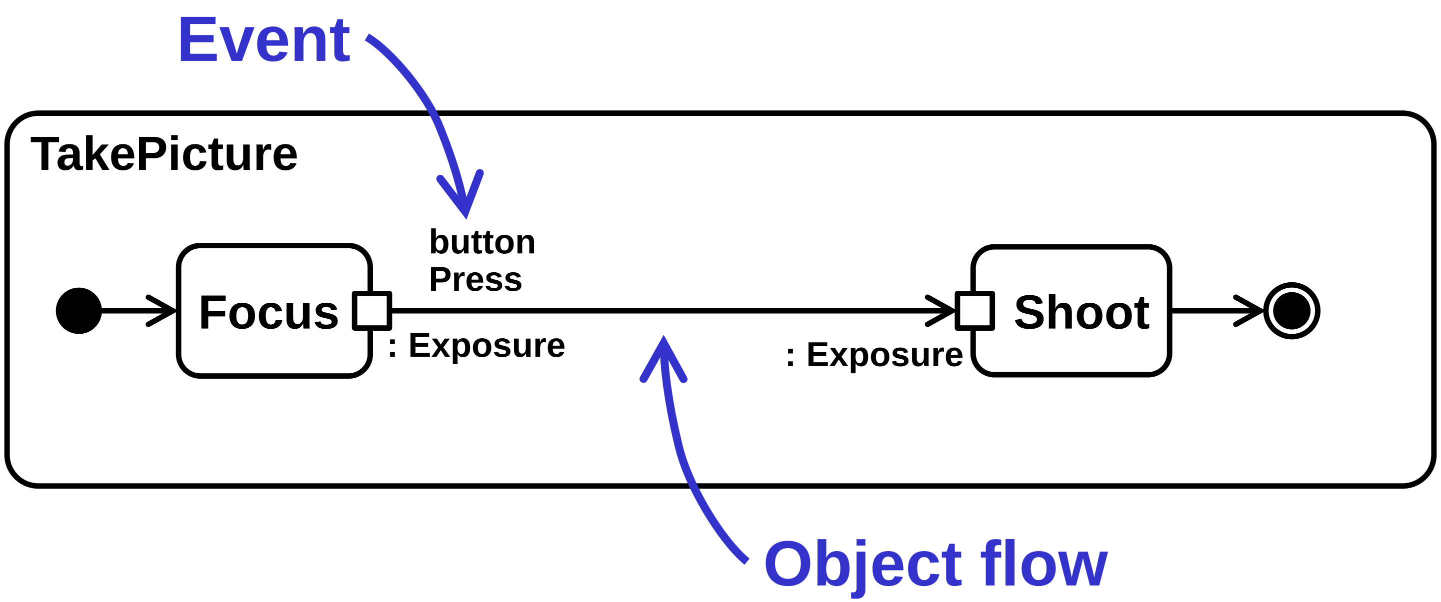
- Flexibility in using metamodels
  - Add metaelements as needed to simplify library usage.
- Many metaelements become synonyms
  - Application / method / diagram-specific terminology sharing same semantics.
  - M2 actions, states, etc, => M1 happensDuring
- Learning UML and building analysis tools for 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, etc)
  - 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 the opposite
  - Tells whether an existing behavior occurrence conforms to a model.
  - Doesn't say how to create an occurrence.
    - Execution engines and reasoners do this.
  - Enables semantic conformance testing.

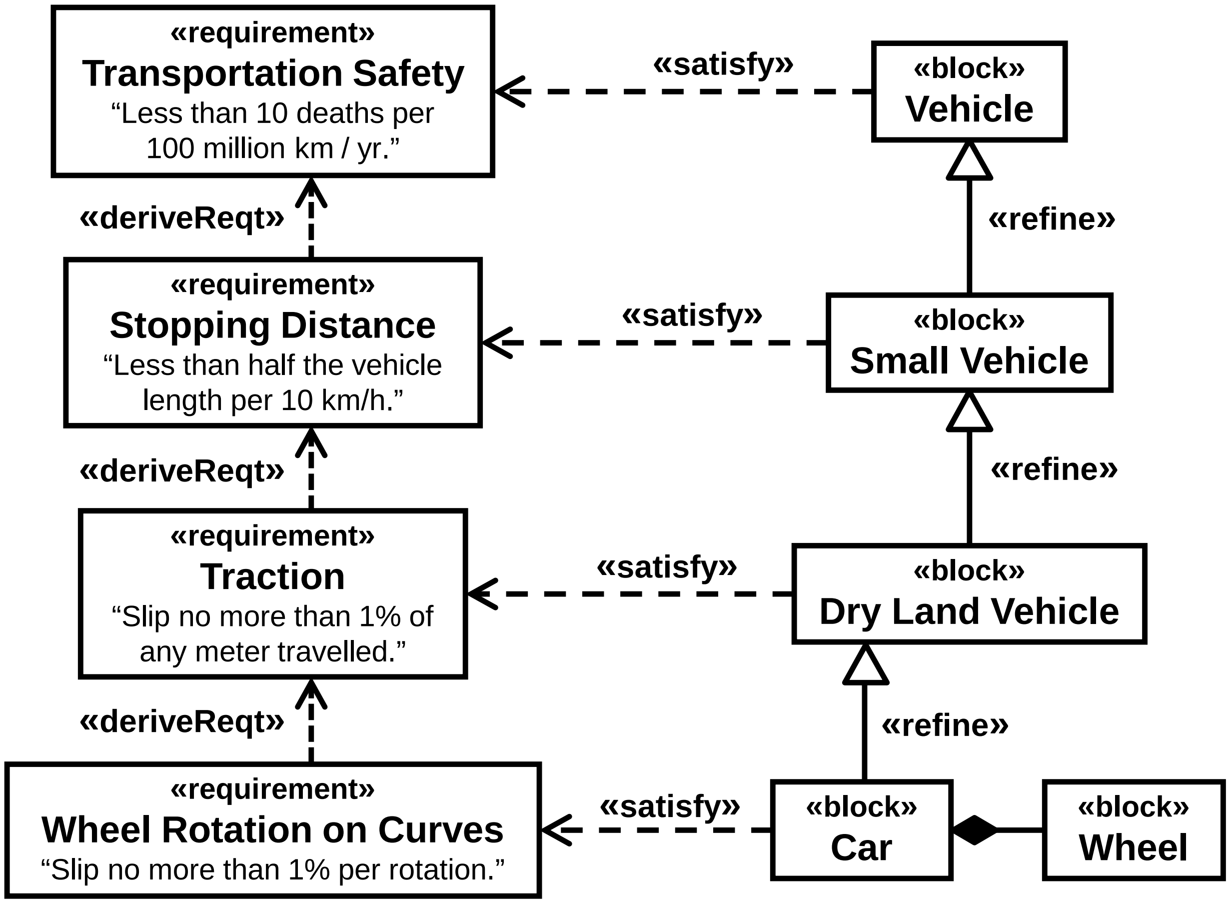
Overview

- RoadMap
- Motivation
  - Behavior, review
  - **Requirements and designs, requirements**
- Requirements and designs, Solution
  1. Satisfaction, derivation, and refinement
  2. Operation and effect
- Summary

SysML background

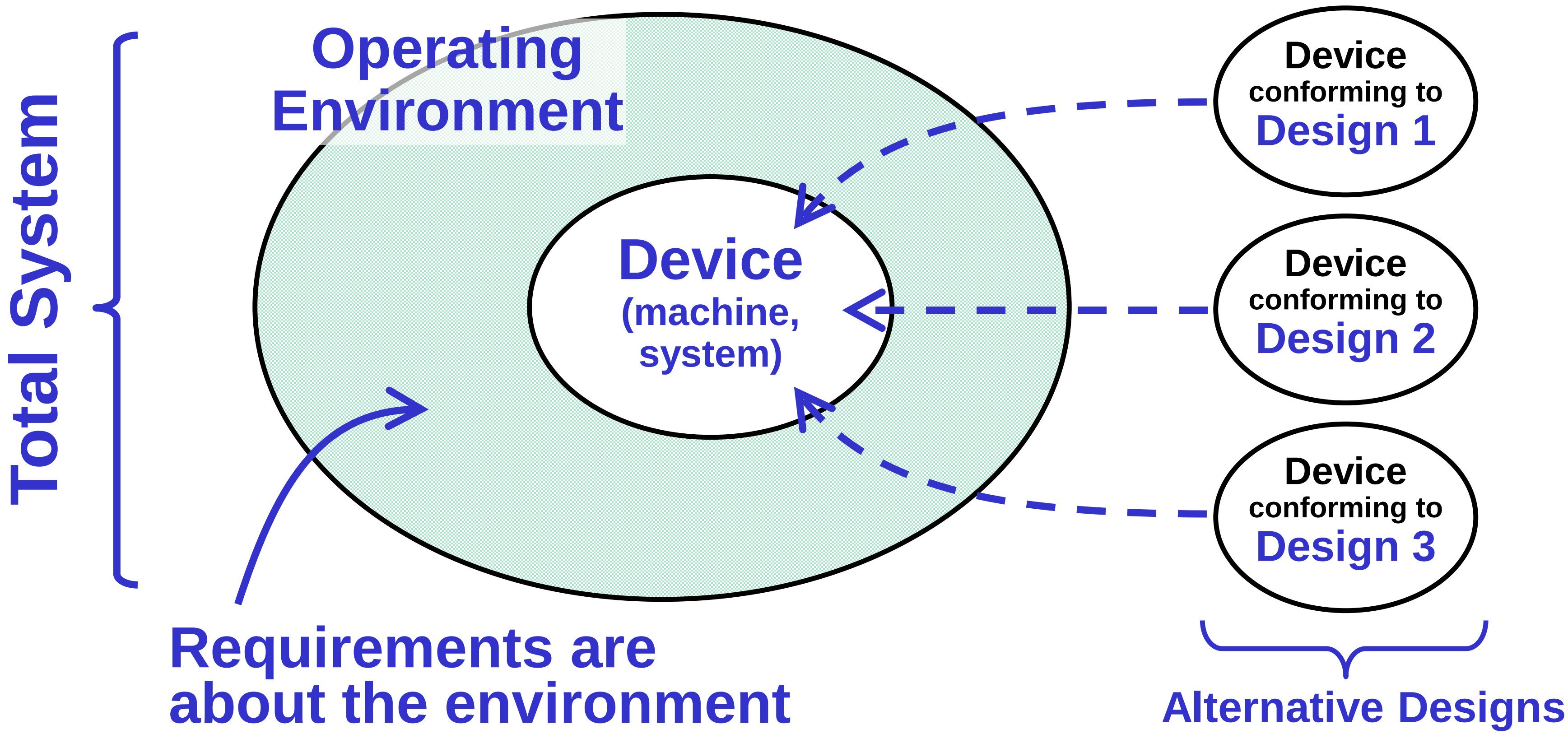
- **Solicited feedback from requirement engineers.**
  - **They couldn't agree.**
- **Only the most widely used capabilities supported in SysML.**
- **SysML (mostly) and this deck about functional requirements**
  - **Functional = during operation.**
  - **Non-functional, eg, cost, manufacturing speed, etc.**

Requirements and Designs



- **Requirements “rolldown”**
  - **Derived alongside design specialization.**

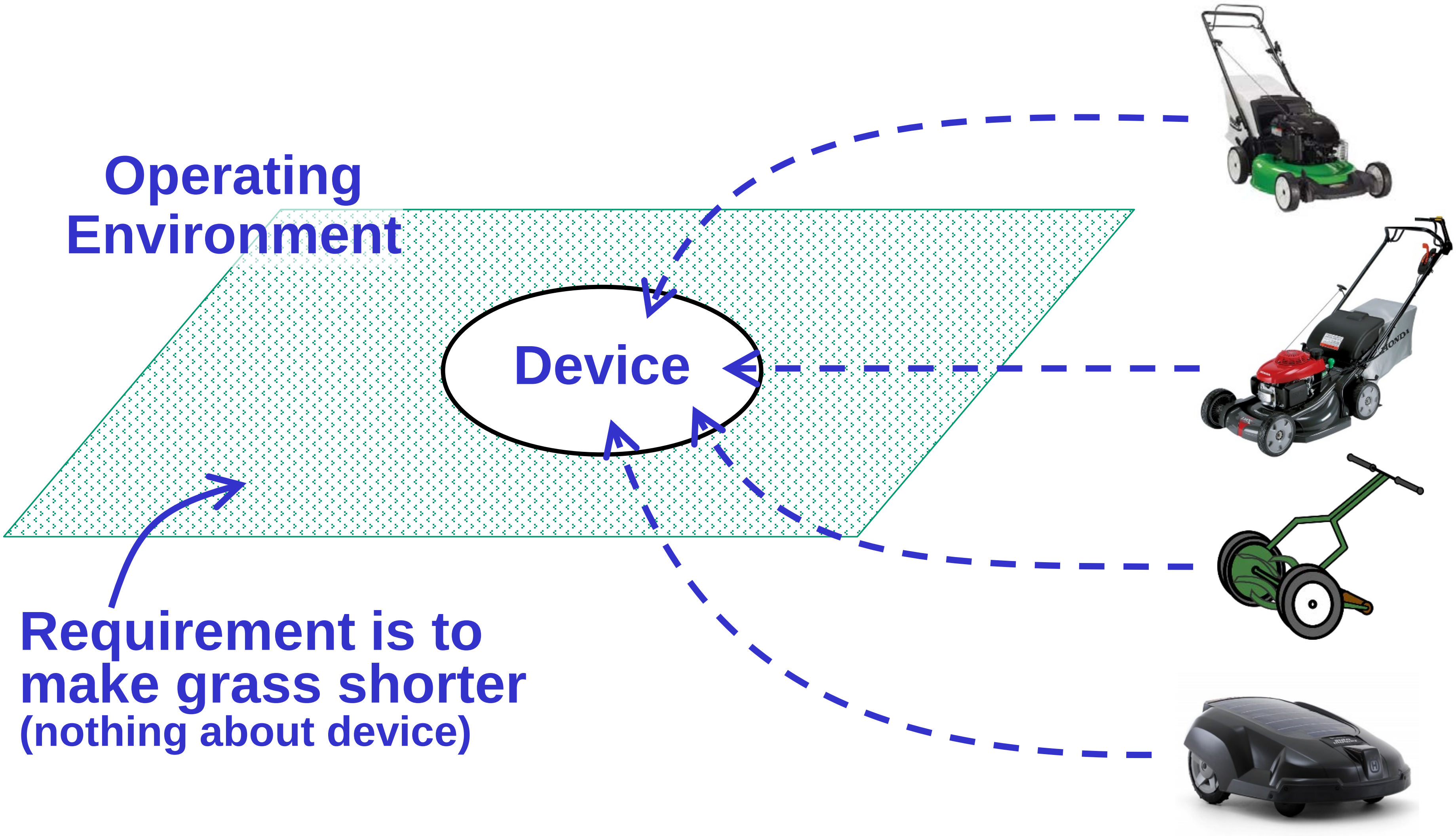
Onto: What's Being Modeled?



- **Requirements specify (constrain) objects around a (hypothetical) device during its operation.**
  - **They do not specify / constrain the device.**

Zave, P., Jackson, M., "Four Dark Corners of Requirements Engineering." ACM Transactions on Software Engineering and Methodology 6 (1): 1–30, 1997.

Lawn Mowing Requirements



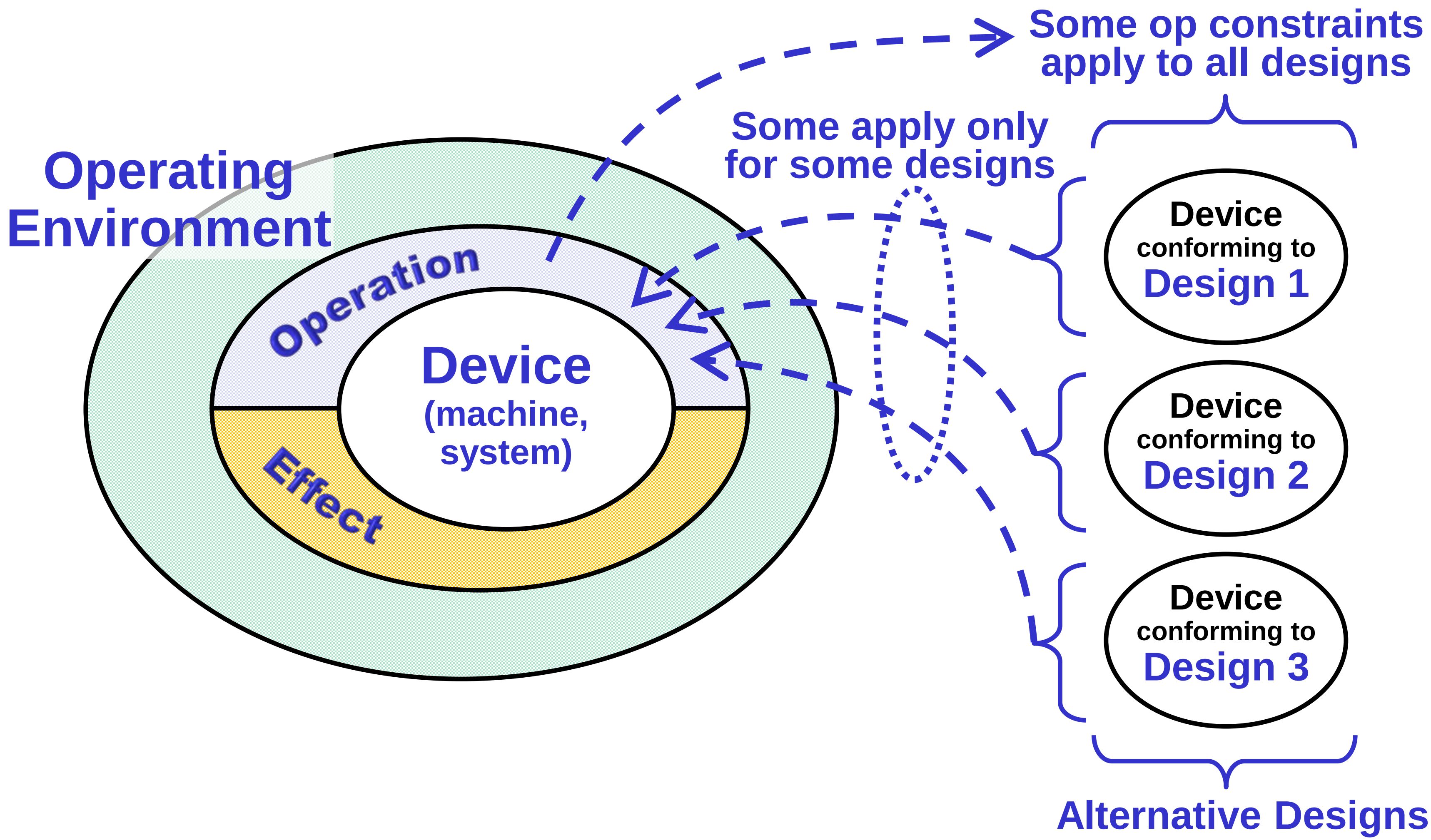
Operating Environment

- Requirements specify:
  - Effects of a (future) device …
  - … when it is operated properly.
- Effects don't matter when device is operated improperly.
  - They could be the right ones or not.

Proper Operation Spec'd By

- Customers, for all (future) designs
  - Eg, easy to use.
  - Limits designers.
- Designers, for their particular design
  - Eg, operating manuals / trainings.
  - Limits operators.
  - Operation differs by kind of device (design)
    - But is still only about environment, not device.
    - Not designs (could apply to multiple designs).

Operation and Effect



- **Want devices that produce the desired effects when operated properly.**

Requirements & Designs Requirements

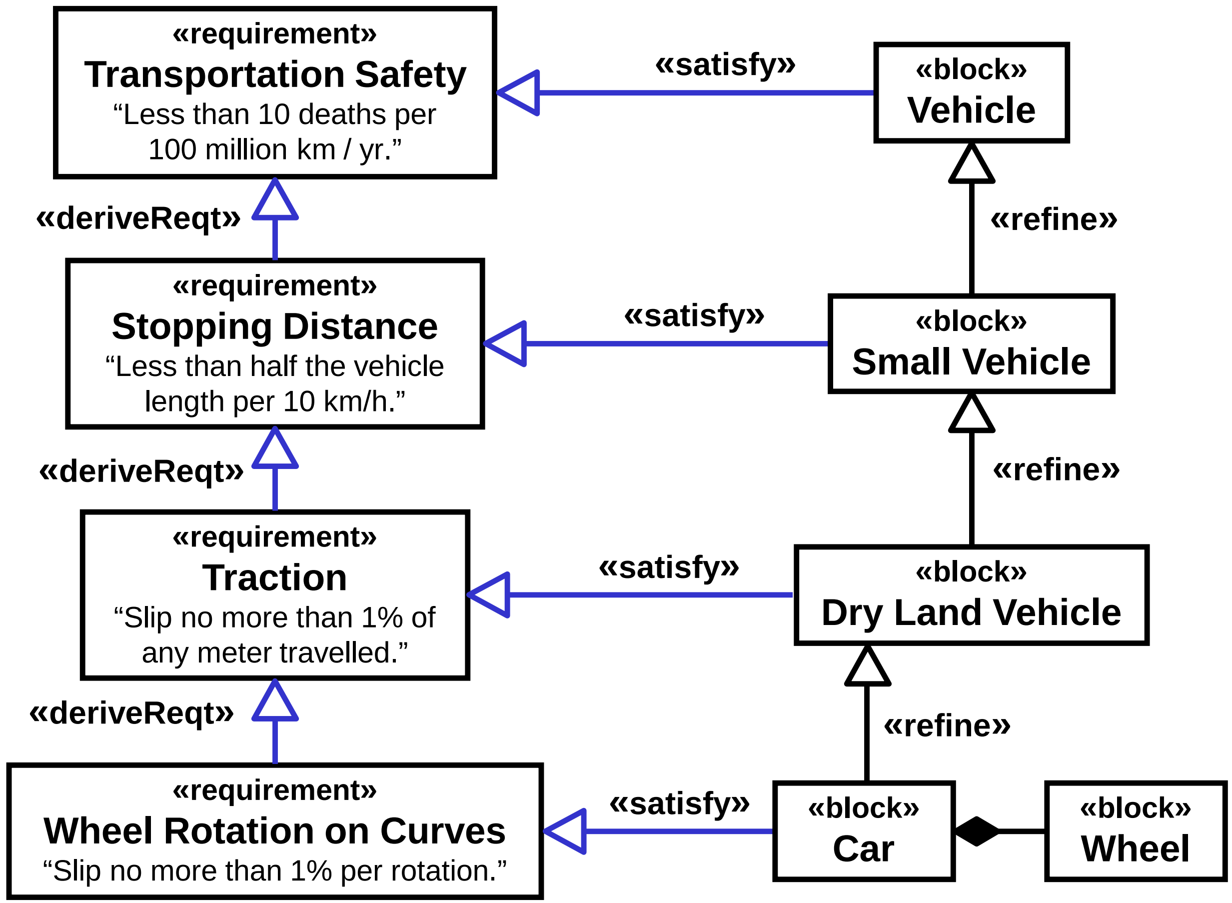
1. Logical interpretation for requirement satisfaction/derivation and design refinement.
2. Treat requirements as specifying operational environment behavior
   – Operation
   – Effect

Overview

- RoadMap
- Motivation
  – Behavior, review
  – Requirements and designs, requirements
- **Requirements and designs, Solution**
  1. **Satisfaction, derivation, and refinement**
  2. Operation and effect
- Summary

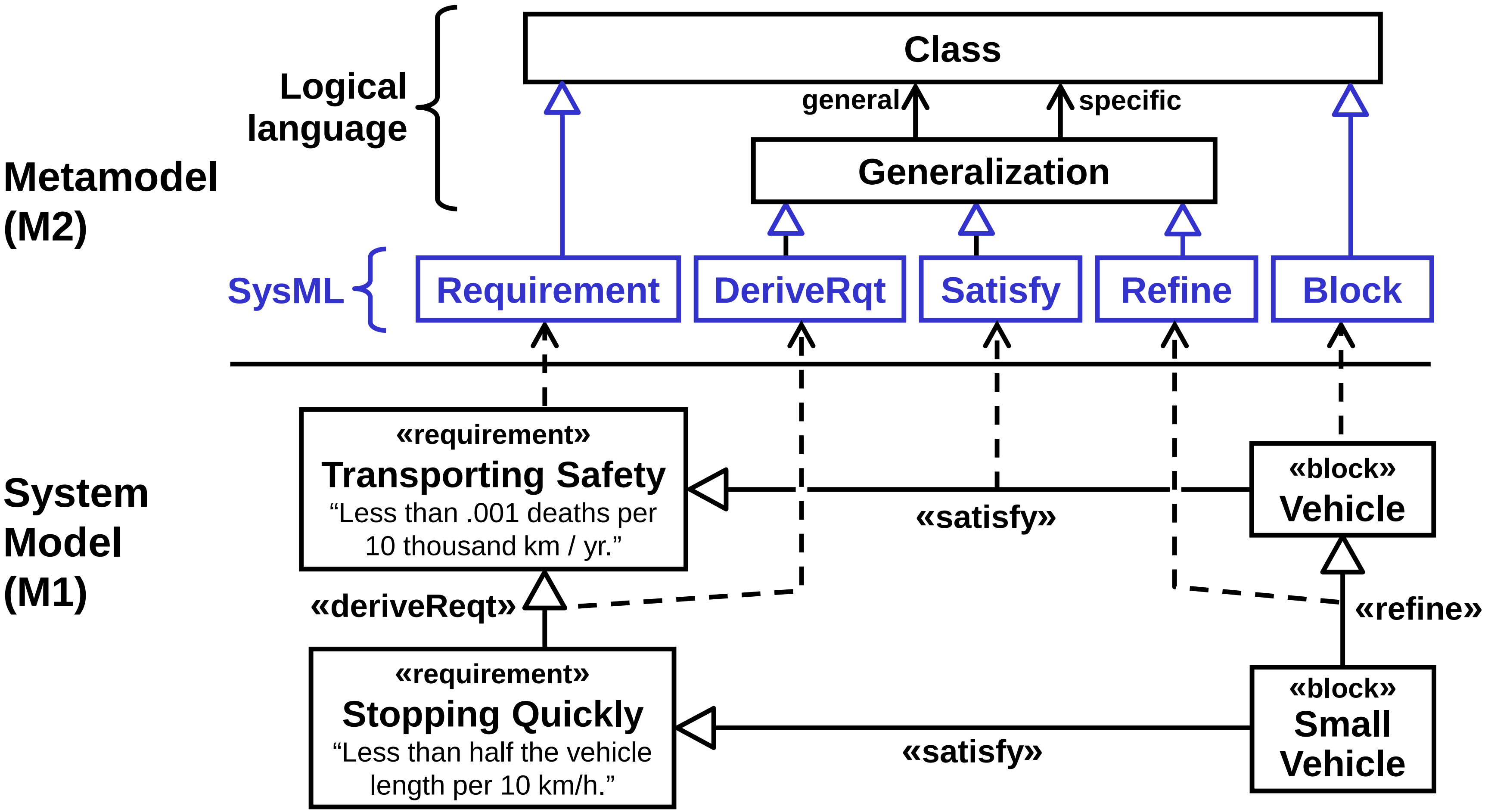
Generalization? (M1)

System Model (M1)



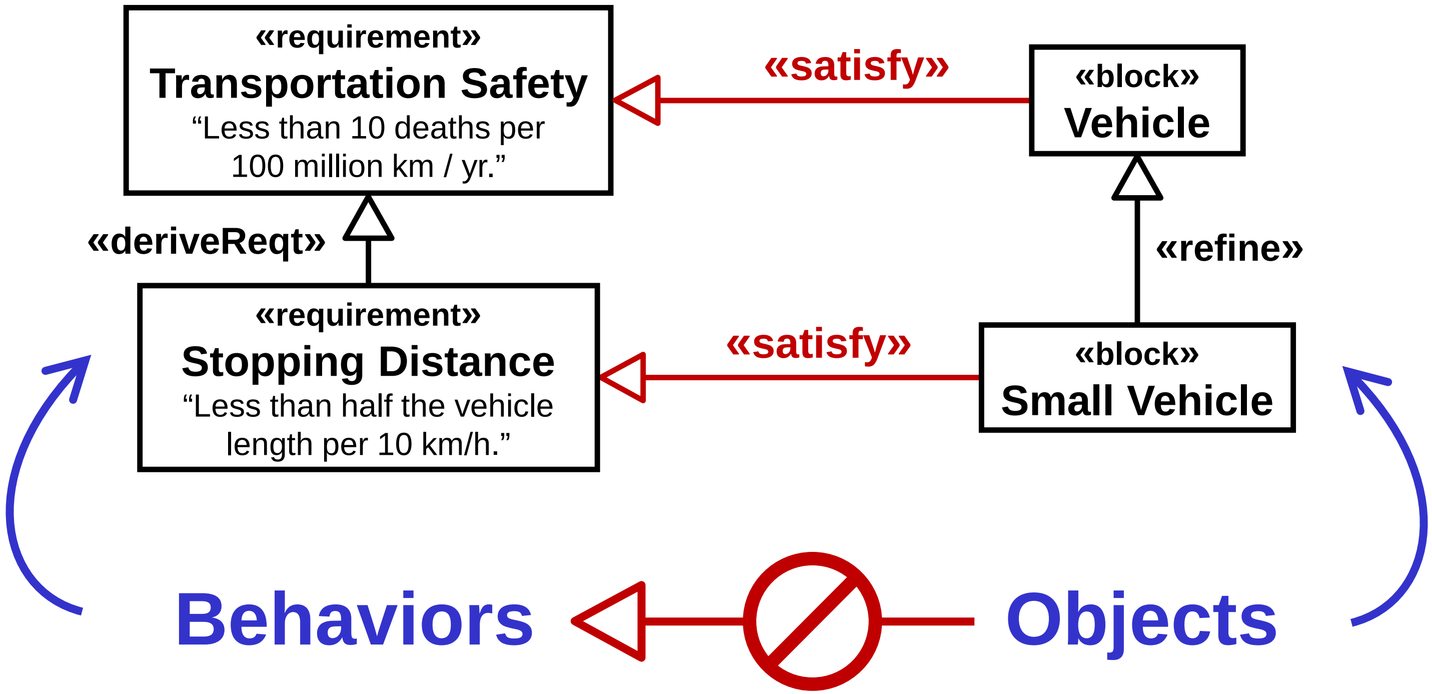
- Designs obey constraints of requirements.
- Derived requirements obey general ones.
- Refined designs obey general ones.

Generalization? (M2)



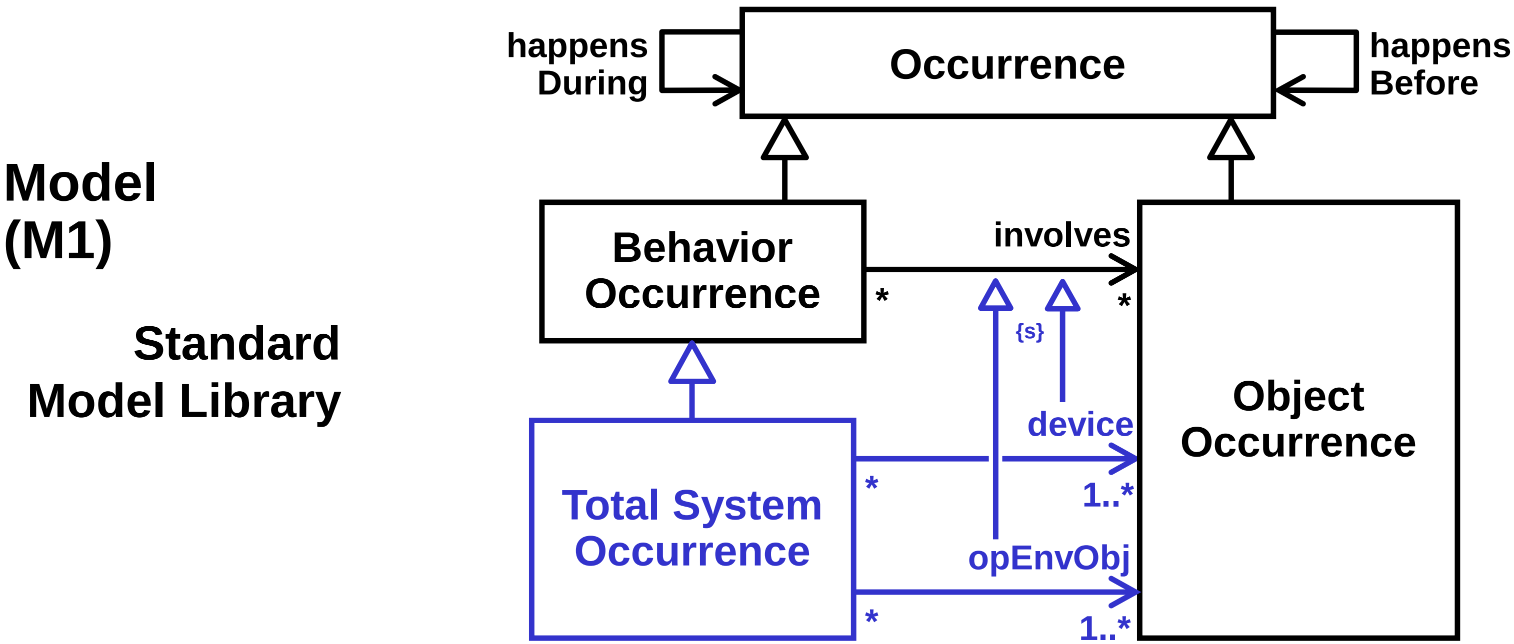
- Engineers use SysML layer of M2
- Reasoners use logical layer

Objects aren’t Behaviors



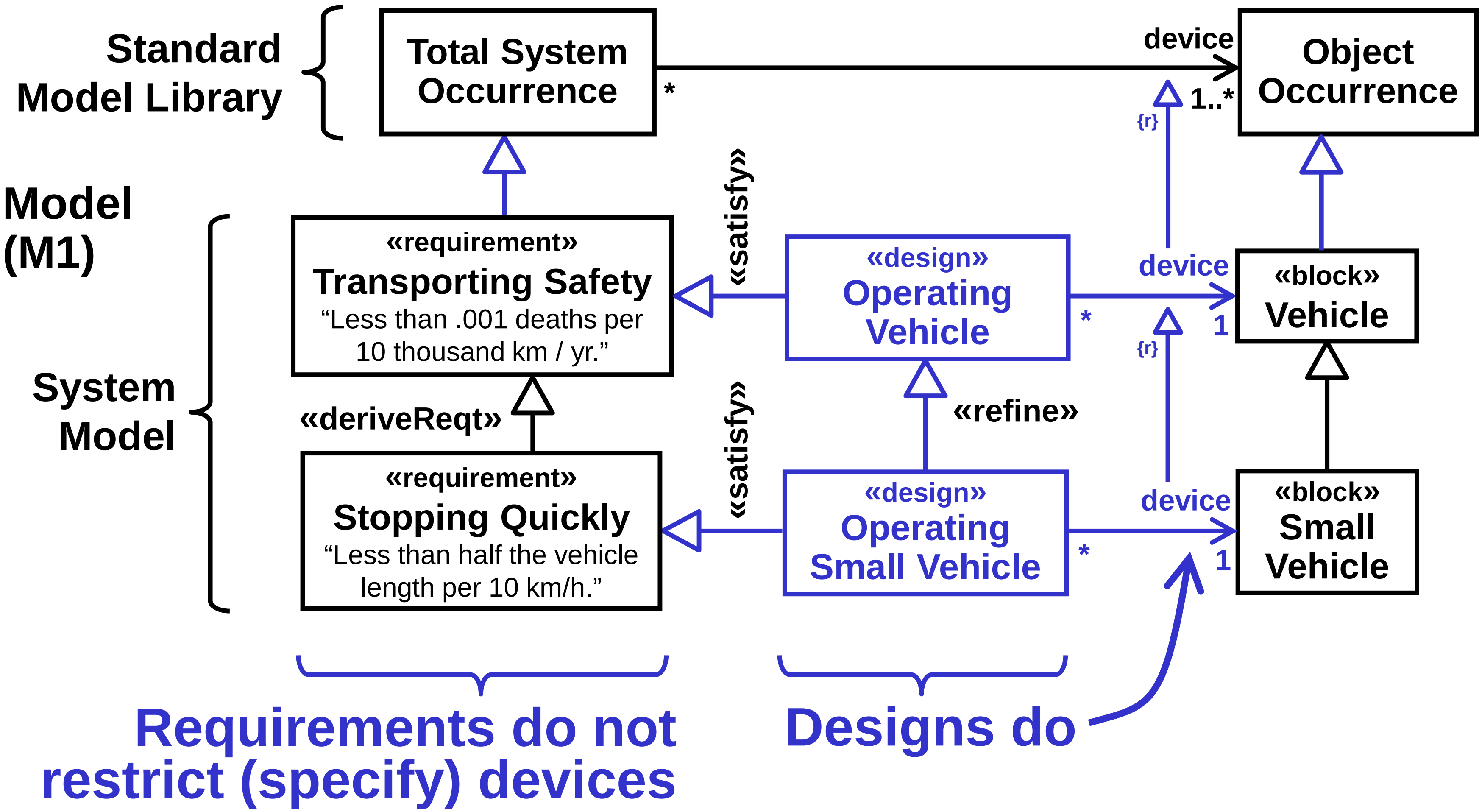
- **Behaviors don’t generalize objects.**
  - **Need same kinds of things on both ends of requirement satisfaction.**

Total Systems



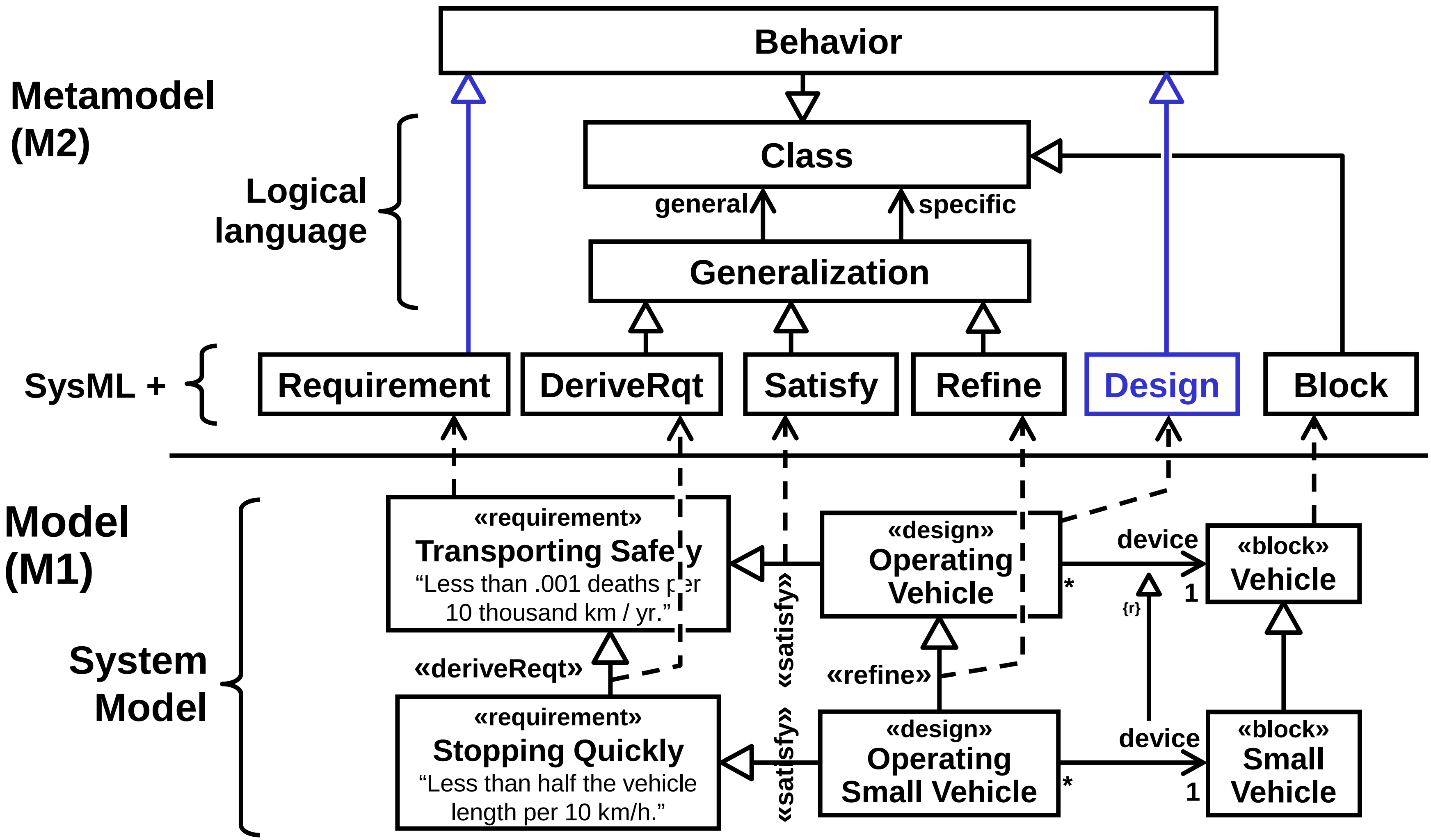
- Behaviors involve objects.
- Total systems involve devices and objects in their environment.

Req/Des as Total Systems (M1)

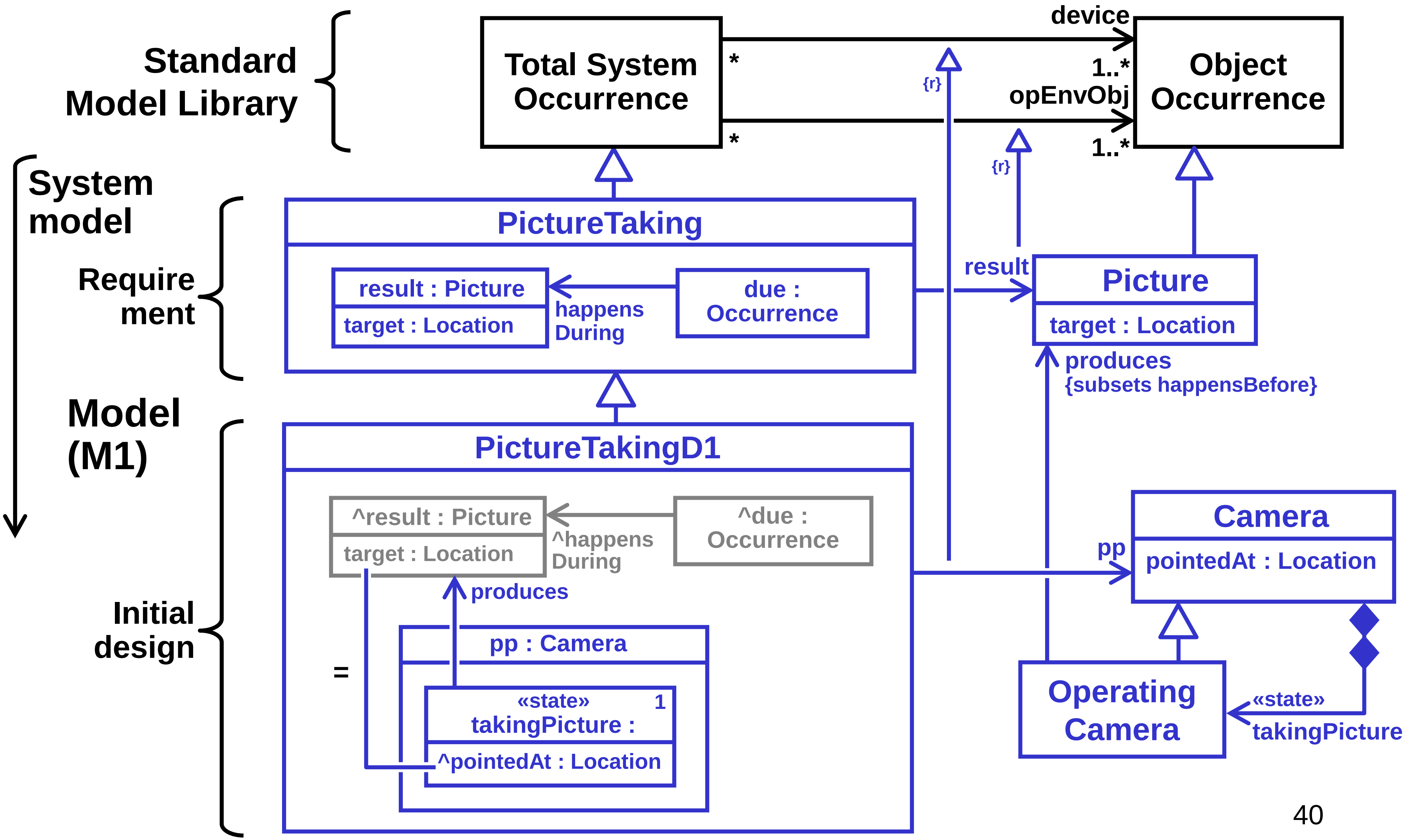


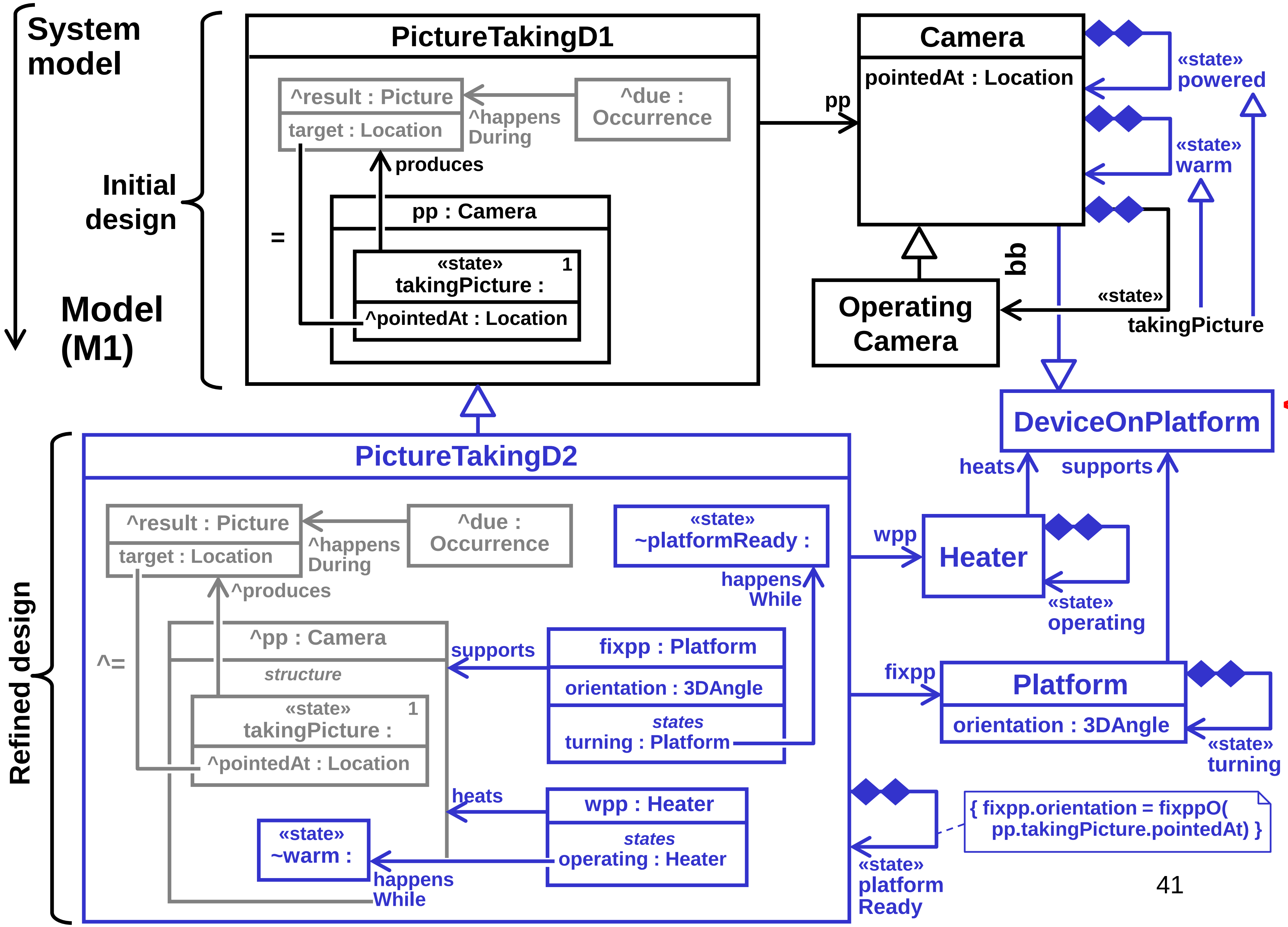
- **Total systems enables generalization to be used for requirement satisfaction.**

Req/Des as Total Systems (M2)

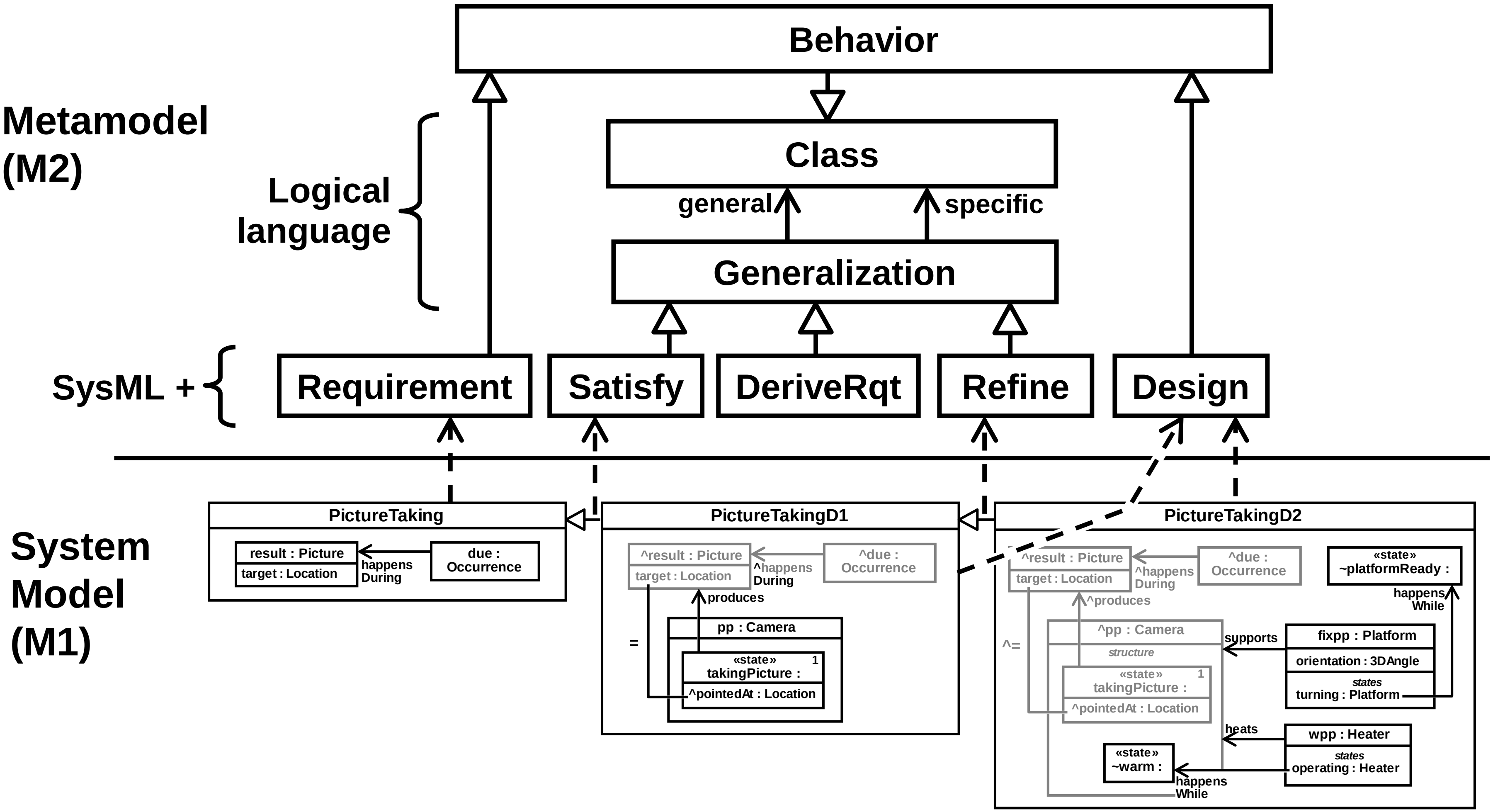


Example Using Generalization (M1)




System model
Initial design
Model (M1)
Refined design
PictureTakingD1
^result : Picture
target : Location
^happens During
^due : Occurrence
produces
=
pp : Camera
«state» takingPicture : 1
^pointedAt : Location
Camera
pointedAt : Location
pp
«state» powered
«state» warm
bb
Operating Camera
«state» takingPicture
DeviceOnPlatform
PictureTakingD2
^result : Picture
target : Location
^happens During
^due : Occurrence
^produces
^=
^pp : Camera
structure
«state» takingPicture : 1
^pointedAt : Location
«state» ~platformReady :
happens While
supports
fixpp : Platform
orientation : 3DAngle
states
turning : Platform
heats
wpp : Heater
states
operating : Heater
«state» ~warm :
happens While
wpp
Heater
«state» operating
heats
supports
fixpp
Platform
orientation : 3DAngle
«state» turning
«state» platform Ready
{ fixpp.orientation = fixppO(pp.takingPicture.pointedAt) }

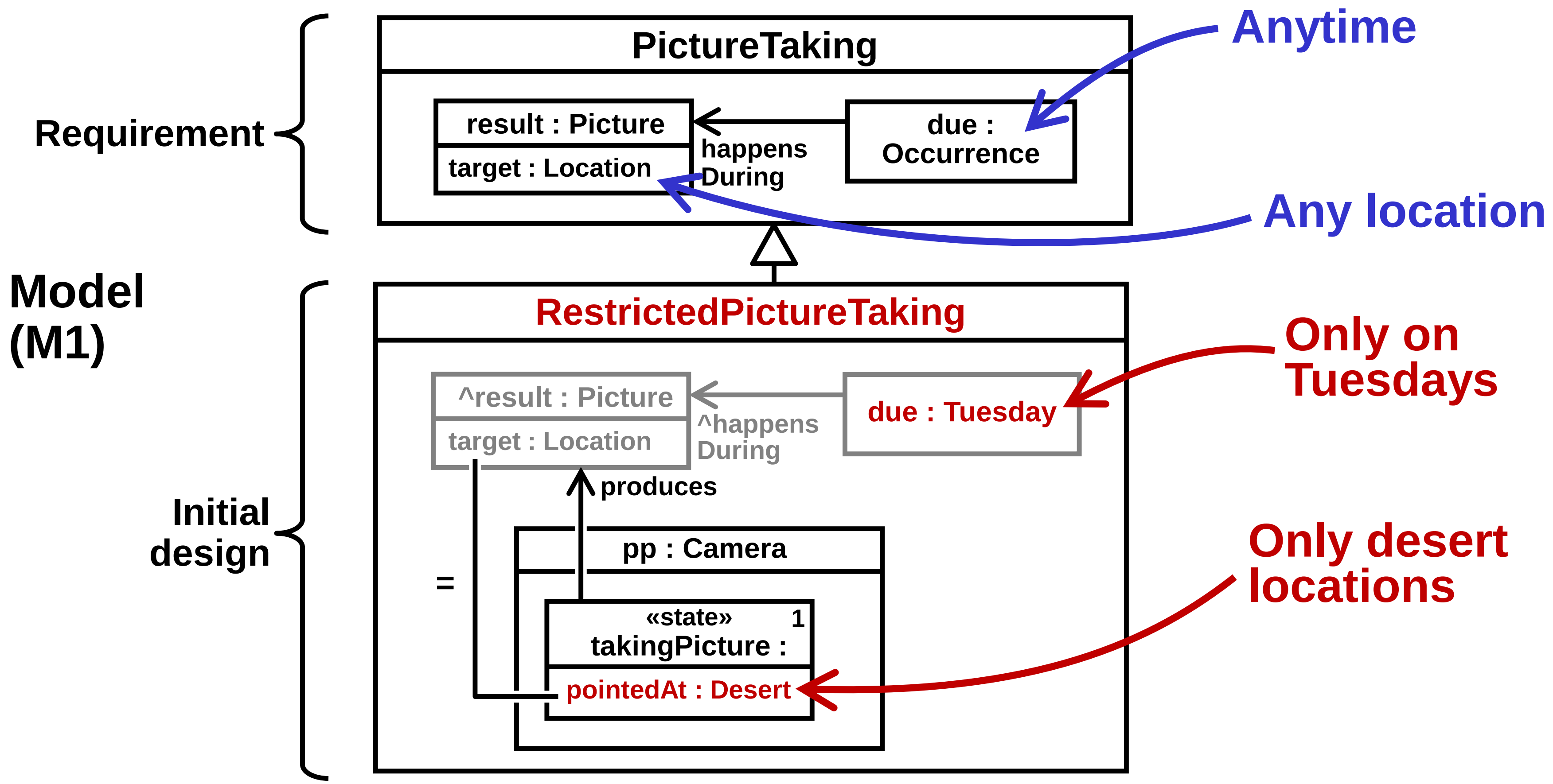
Example Using Generalization (M2)



Generalization? Not quite.

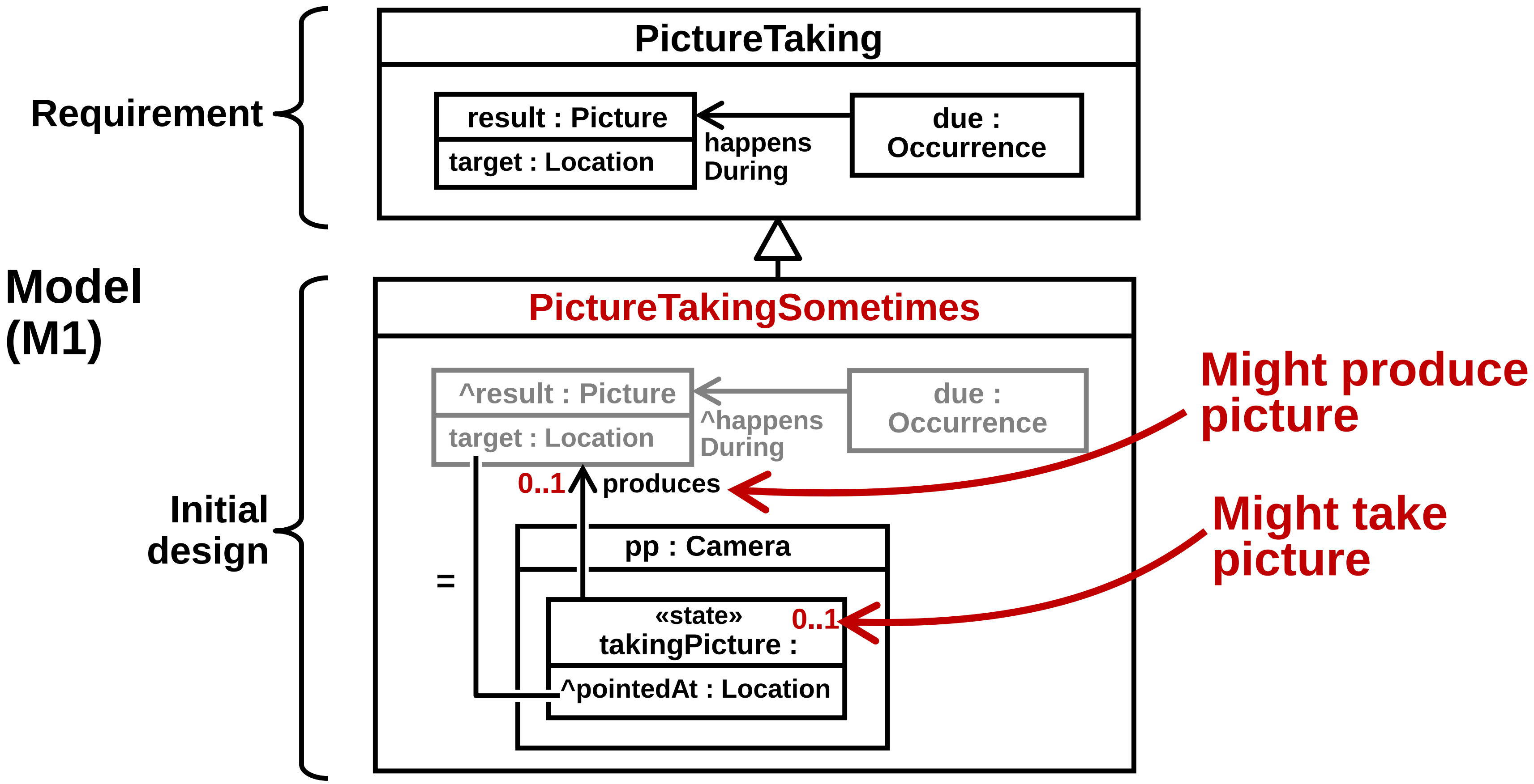
1. Design can restrict operation requirement
   – Design could work in limited operation cases and still be a specialization.
2. Design will always satisfy effect requirement
   – Design occurrences that don't have desired are excluded by generalization.

1) Design Restricting Operations



- Specializations are usually narrower.
  - = subsets of PictureTaking occurrences.

2) Effects Always Satisfied



- Specializations might loosen effects
  – But still subsets PictureTaking occurrences.

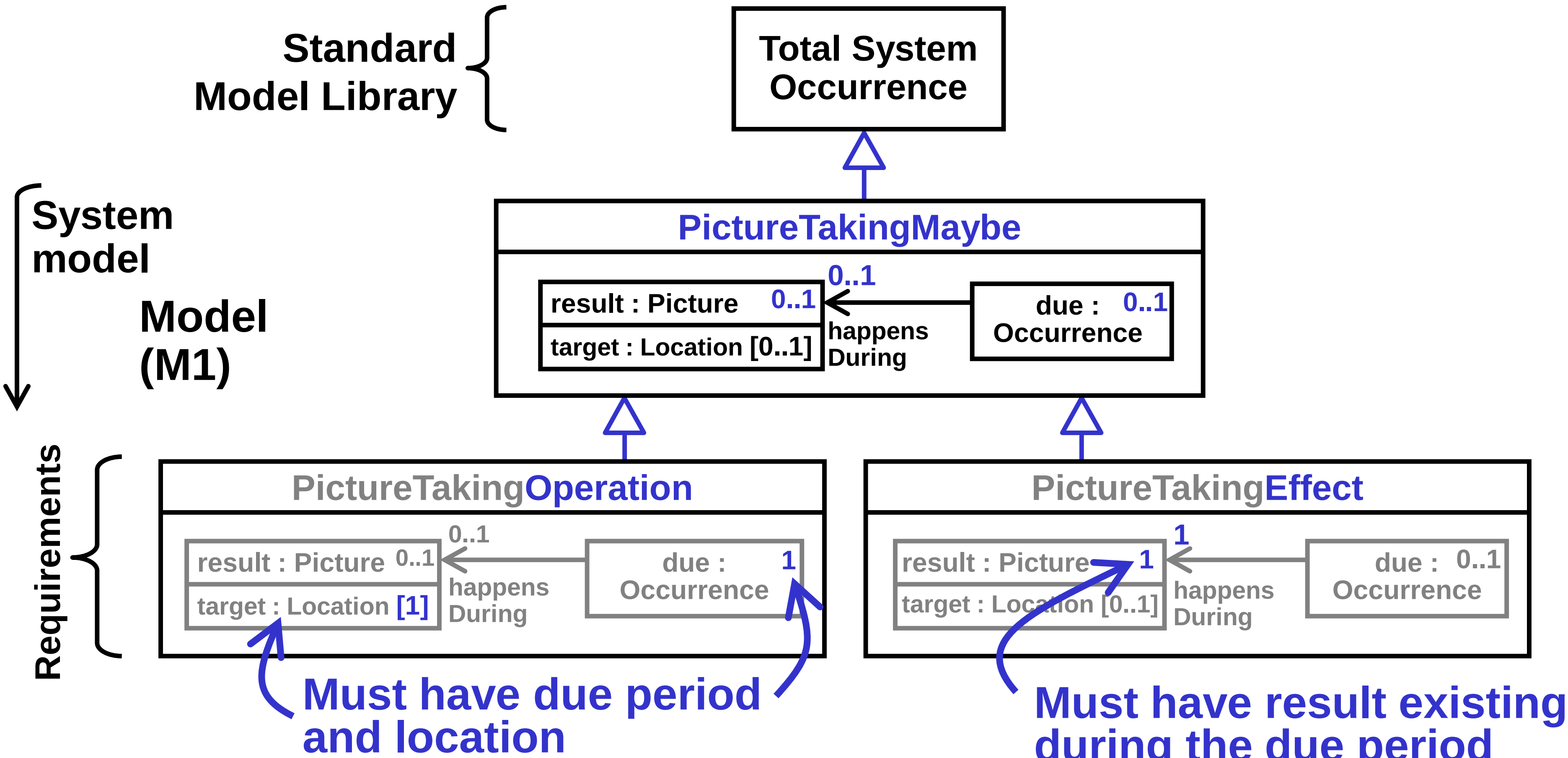
Syntactic Solutions? Not quite.

1. UML::isFinal = true on operation elements prevents narrowing type/mult
   - Doesn't prevent restrictions due to
     - Bindings (eg, to camera target).
     - Interactions among design elements.
2. UML redefinition doesn't allow loosening type/mult (on effect elements)
   - Doesn't prevent loosening due to
     - Decision nodes / states, alt fragments.
     - Interactions among design elements.

Overview

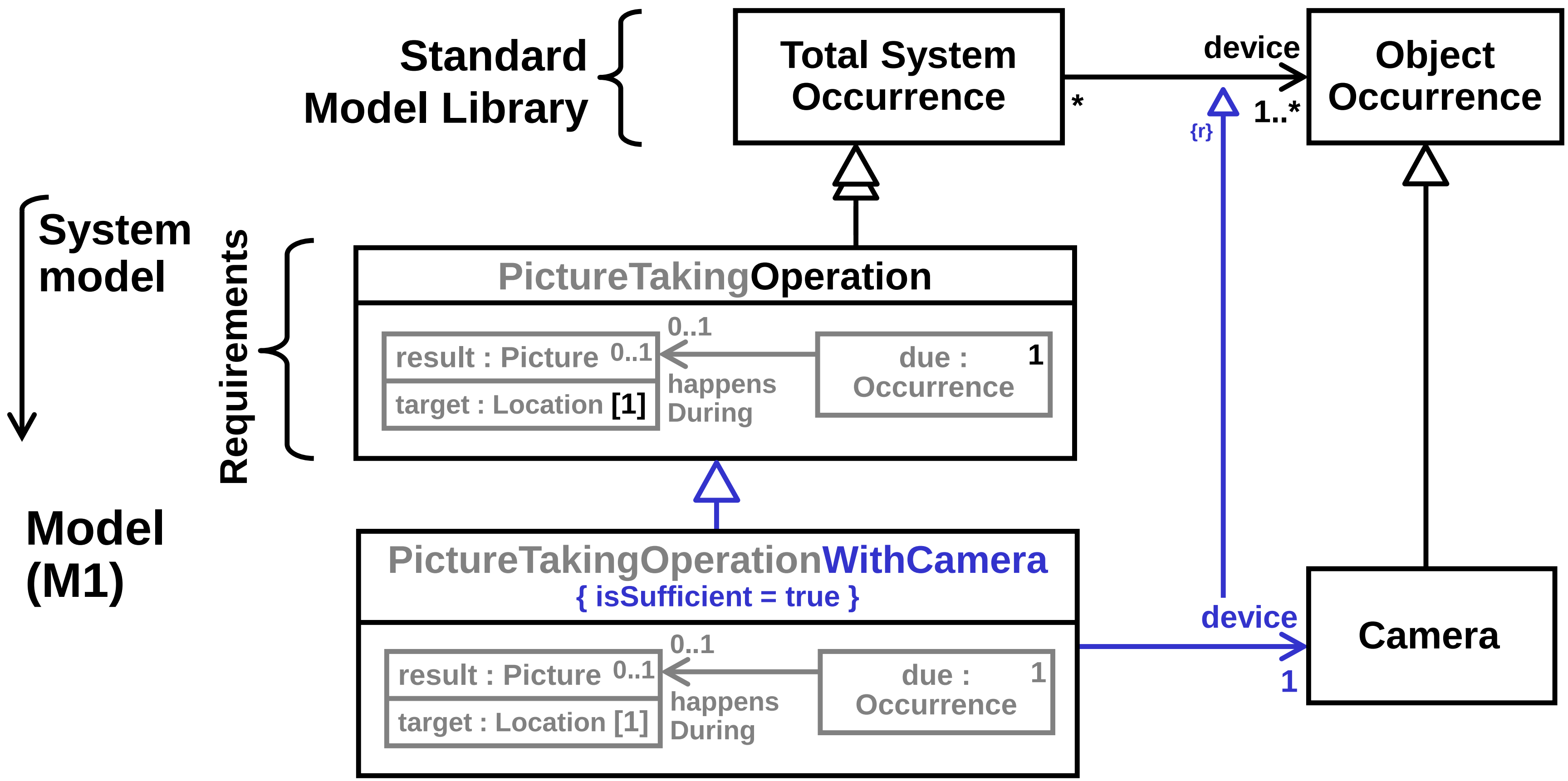
- RoadMap
- Motivation
  – Behavior, review
  – Requirements and designs, requirements
- Requirements and designs, Solution
  1. Satisfaction, derivation, and refinement
  2. **Operation and effect**
- Summary

Separate Operation And Effect



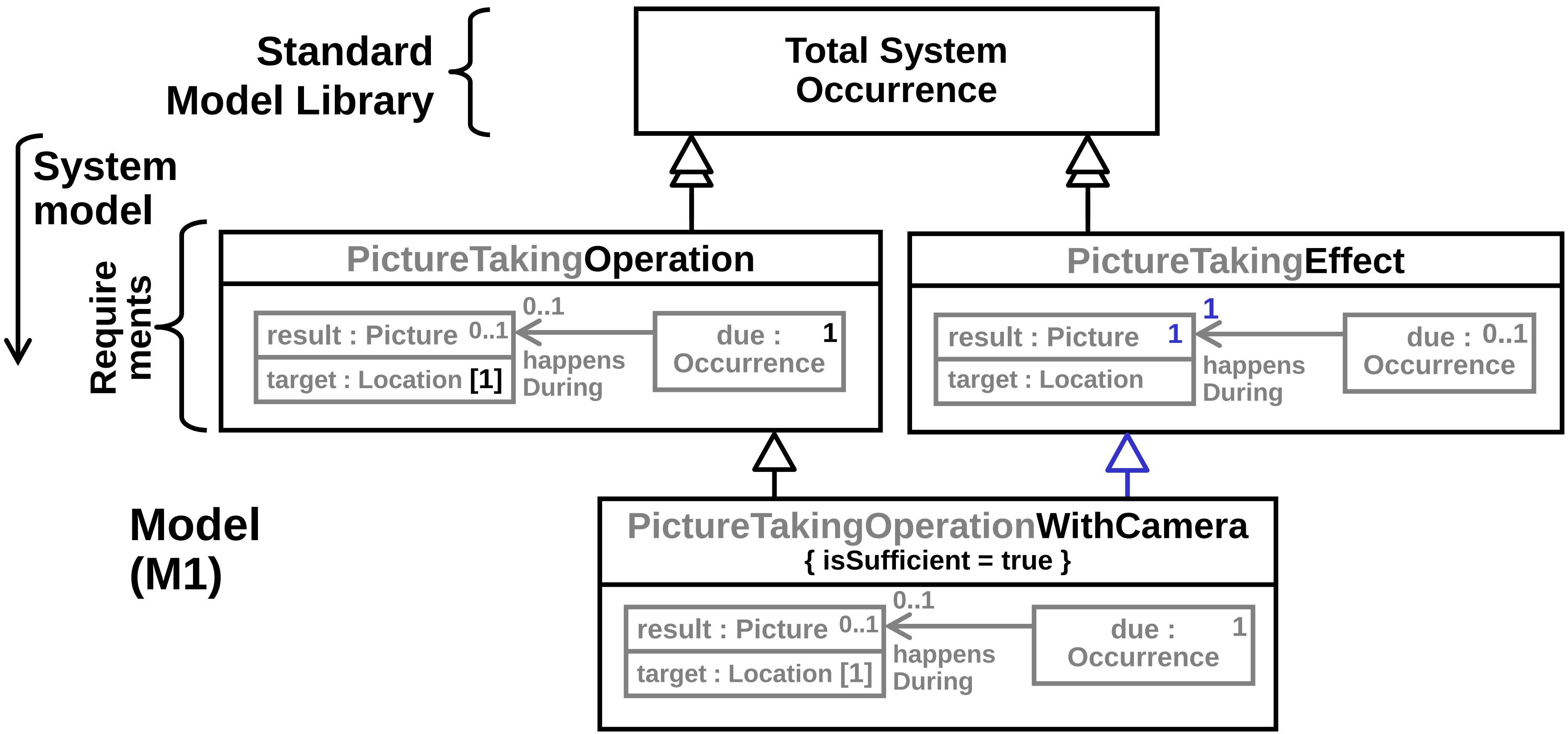
- **Distinguished which parts of requirement are mandatory.**

All Operations of Device …

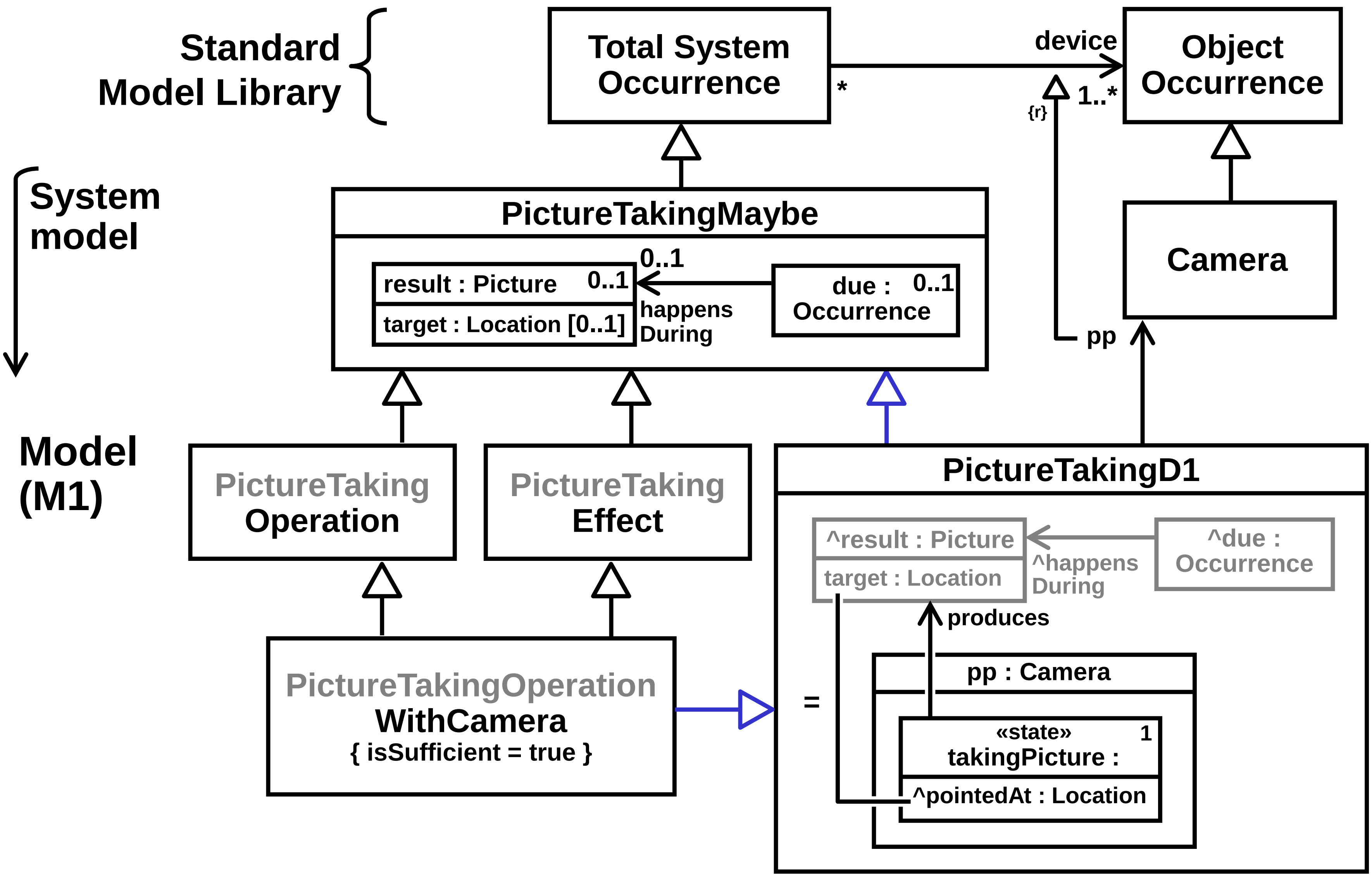


- **All total system occurrences satisfying operation requirements on using device**
  - **Anytime, any location.**

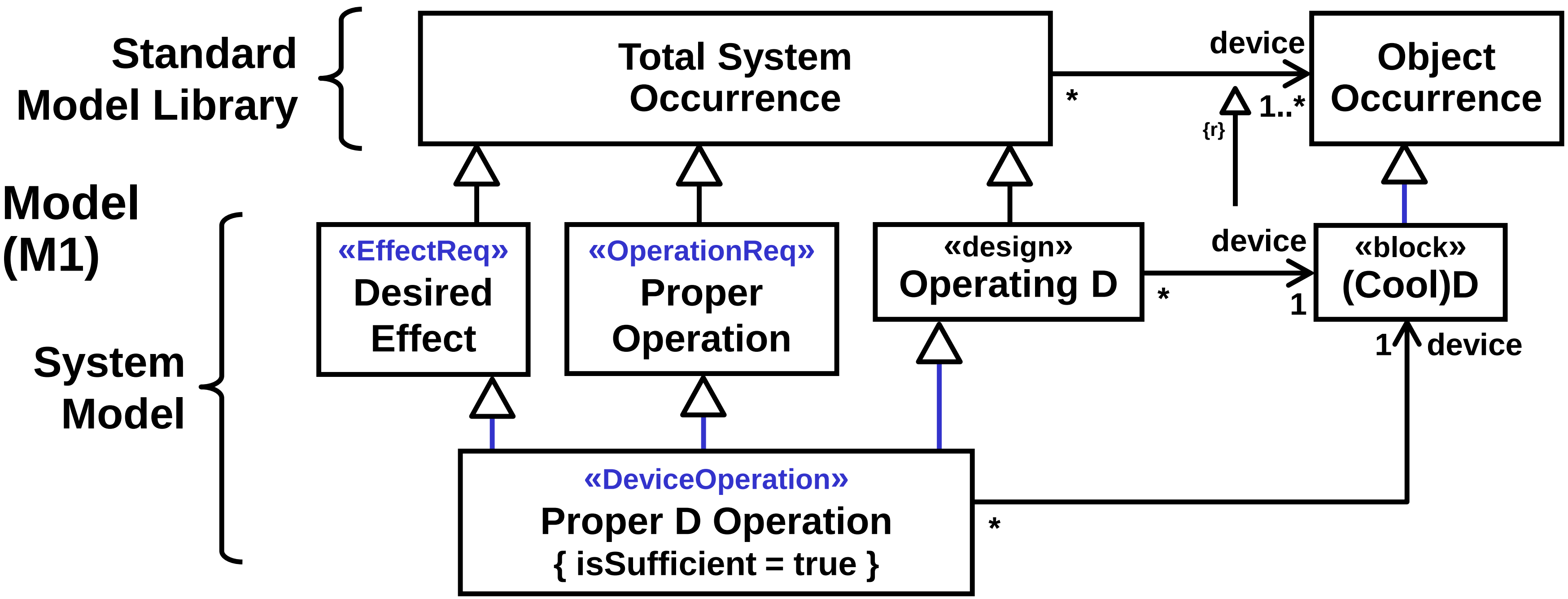
... Have Desired Effects ...



… And Conform to Design



Req Satisfaction Pattern (M1)



- Stereotypes indicate possible M2
  - TBD

TBD

- **Easier requirement satisfaction modeling**
  - **M2**
  - **Anything simpler for M1?**
- **Finding requirement violations**
  - **Tests**
  - **Search**

Overview

- RoadMap
- Motivation
  – Behavior, review
  – Requirements and designs, requirements
- Requirements and designs, Solution
  1. Satisfaction, derivation, and refinement
  2. Operation and effect
- **Summary**

Summary

- Requirements specify operating environment of device only
  - Operation and effect.
- Designs specify device only
  - Internals and relation to environment.
- Treat requirements and designs as **behavior occurrences**
  - of operating **environment and device together** (total system).

Summary

- Requirement satisfaction is not generalization. Design
  - Must produce desired effects whenever operated properly.
  - Can’t narrow operations.
  - Can’t loosen effects.
- Generalize all operations of a kind of device (sufficient) by
  - Proper operation
  - Desired effect
  - Design of the same kind of device (as total system behavior).

Past ADTF Intro Slides

- **Intro to Behavior as Composite Structure**
  - **http://doc.omg.org/ad/2018-03-02**
- **Interactions: http://doc.omg.org/ad/18-06-11**
- **Object-orientation: http://doc.omg.org/ad/18-09-07**
- **State Machines, parts 1&2:**
  - **http://doc.omg.org/ad/18-12-09**
  - **http://doc.omg.org/ad/19-03-02**
- **Activities, part 2: http://doc.omg.org/ad/19-06-02**
- **4D: http://doc.omg.org/ad/19-09-07**

More Information

- **Earlier slides (more onto)**
  - http://conradbock.org/bock-ontological-behavior-modeling-jpl-slides.pdf
- **Papers:**
  - **Ontological Behavior Modeling:** http://dx.doi.org/10.5381/jot.2011.10.1.a3
  - **Ontological Product Modeling:** https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=822748
  - **4D Requirements Modeling:** https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=919164
- **Application to BPMN:** http://conradbock.org/#BPDM
- **SysML2: Contact Bjorn Cole** bjorn.f.cole@lmco.com