CTL
Communications Technology Laboratory

SST/SysML2 Semantic Assets and Debt : “Event” Handling

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

Ed Seidewitz, Model Driven Solutions
Manfred Koethe, 88 Solutions
Øystein Haugen, Østfold University College

Not responsible for errors

NIST
National Institute of Standards and Technology

Overview

- **Event handling, requirements**
- **Solutions, Kernel**
  - **Onto messages/flows**
  - **Transitions**
  - **Accepting “events”**
- **Solutions, SysML**
  - **Accept and send actions**
  - **Sequence diagrams**
  - **Flows**
- **Summary**

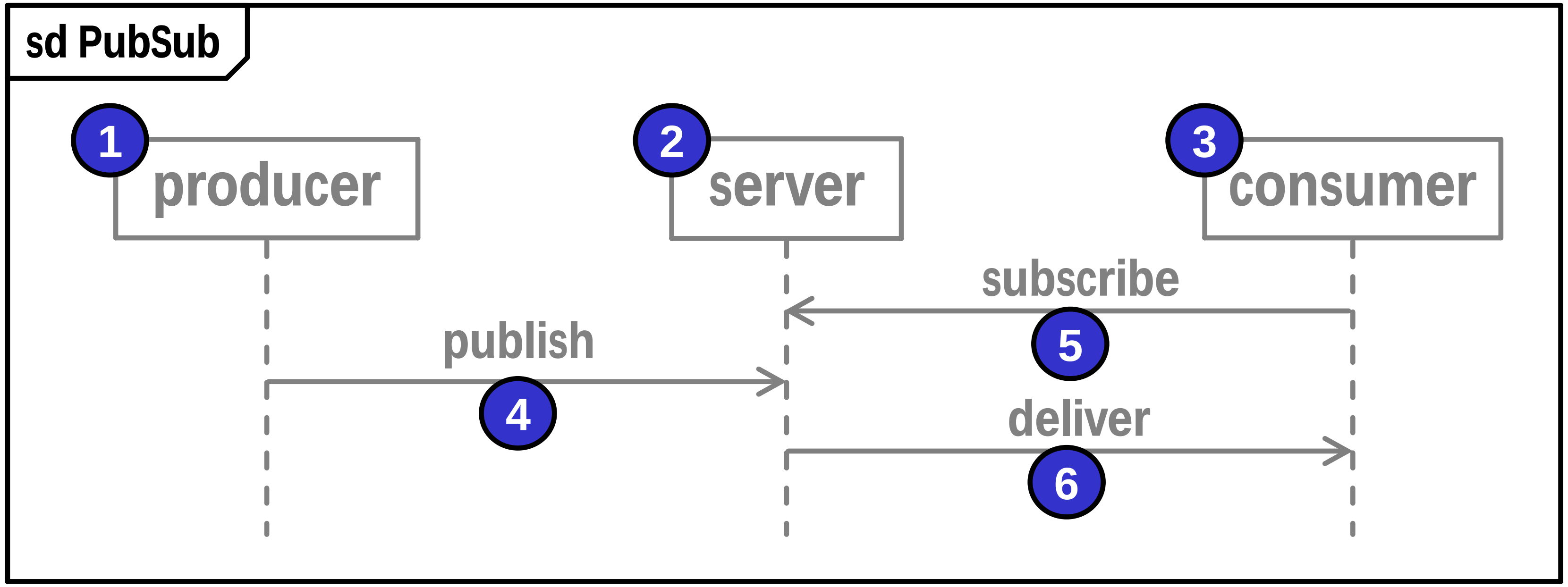
Overview

- **Event handling, requirements**
- Solutions, Kernel
  - Onto messages/flows
  - Transitions
  - Accepting “events”
- Solutions, SysML
  - Accept and send actions
  - Sequence diagrams
  - Flows
- Summary

UML “Event” Handling

- If you don’t know what this is …
  - … you probably won’t need to be bothered with it.
- About objects managing “things” coming at them in concurrent systems (“agent”-like). Often …
  - to honor expectations/agreements between objects about their interactions.
  - managed internally by state machines.

Concurrent Systems

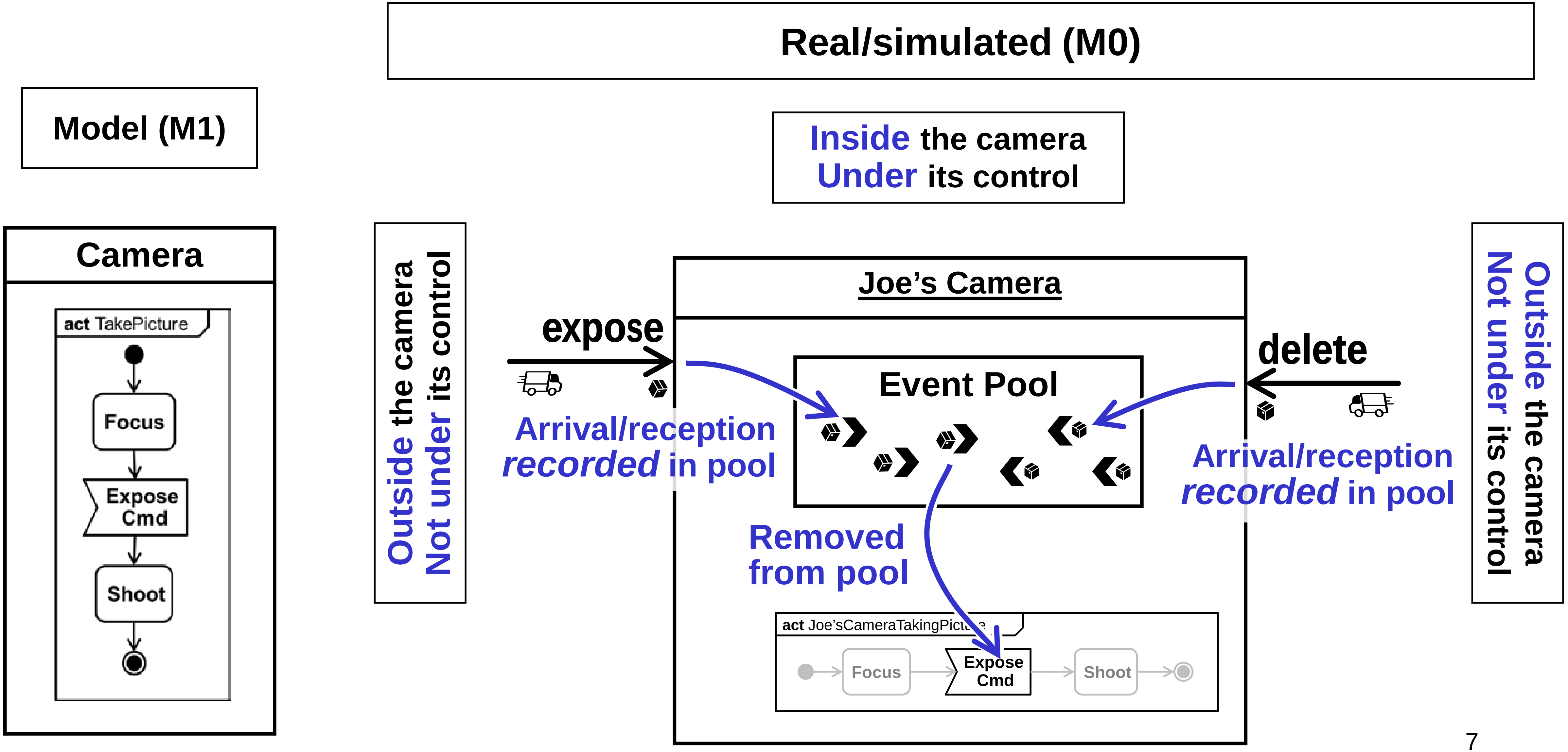


- **Six behaviors (at least)**
  - **Three participants (with internal behaviors)**
  - **Three messages (take time to get there)**

SST-izing UML Event Handling

- “S”-word
  - UML informally describes an event handling procedure.
  - SST requires conditions for checking whether events *were* handled properly (aka “trace” checking, declarative).
- Integration with other SST modeling
  - More flexible event handling than UML.
  - Sequence diagrams
  - Item flows
  - Ports

UML “Events”



Real/simulated (M0)

Model (M1)

Inside the camera
Under its control

Outside the camera
Not under its control

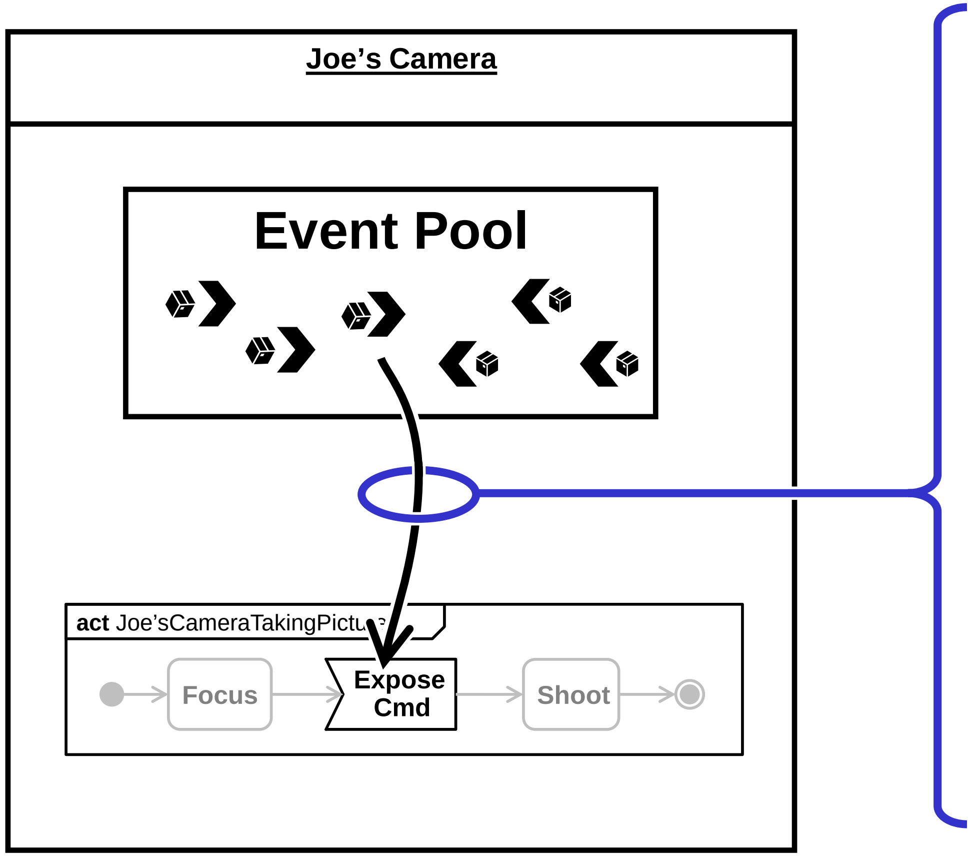
Outside the camera
Not under its control

Arrival/reception
recorded in pool

Arrival/reception
recorded in pool

Removed
from pool

“Processing” Events



1) Select event in pool
   - In specific orders (priority).
   - Remove from pool (dispatch).
     - Deferral (“put back”).

2) Match?
   - Checking required conditions (triggers) against selected event.

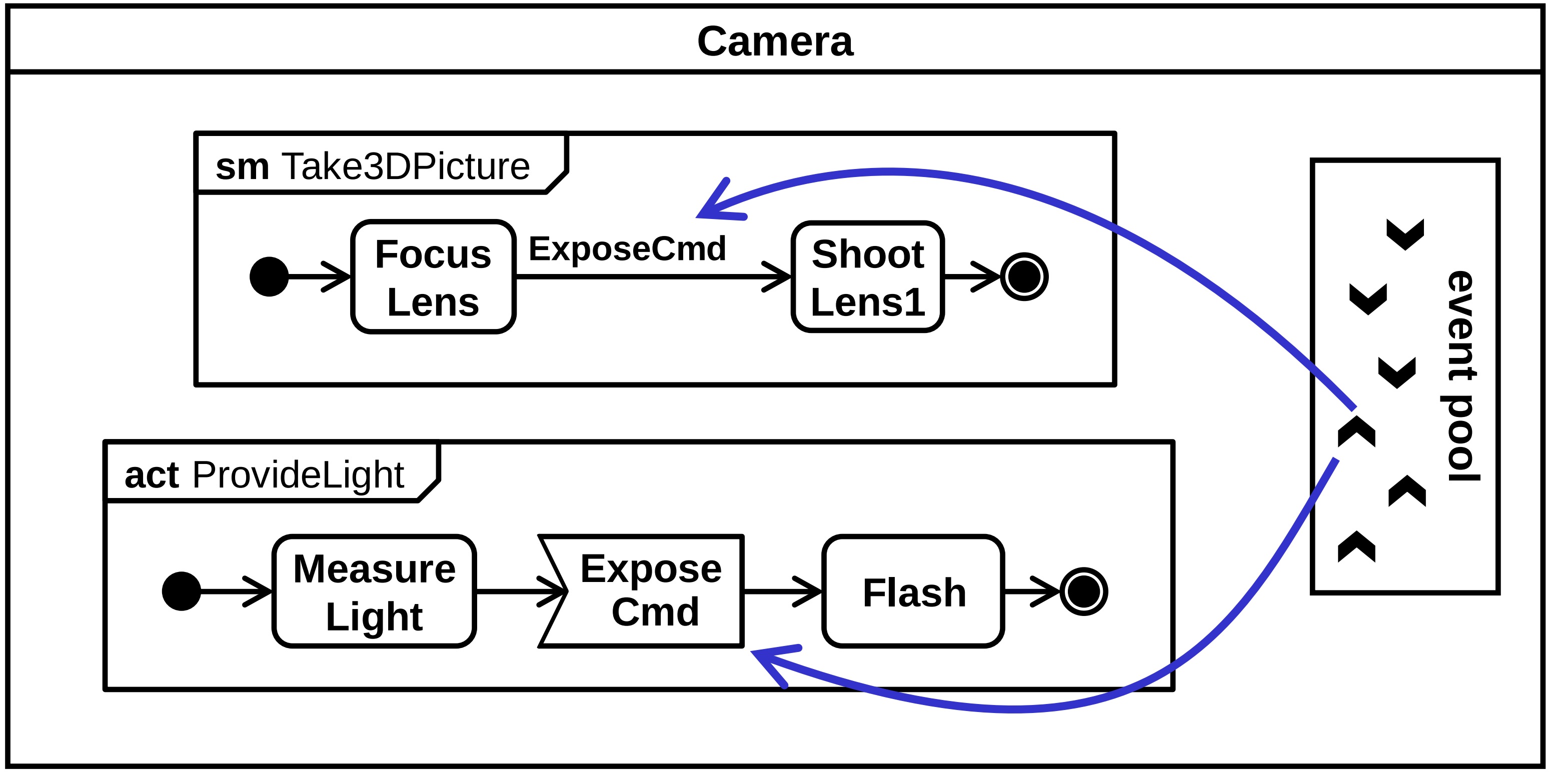
3) (States) Evaluate guards

Removed events might not affect behavior (not “fire”, be “accepted”)

§ Specified as a procedure.
– How to onto-ize it?

SST: Avoid Pool Conflicts

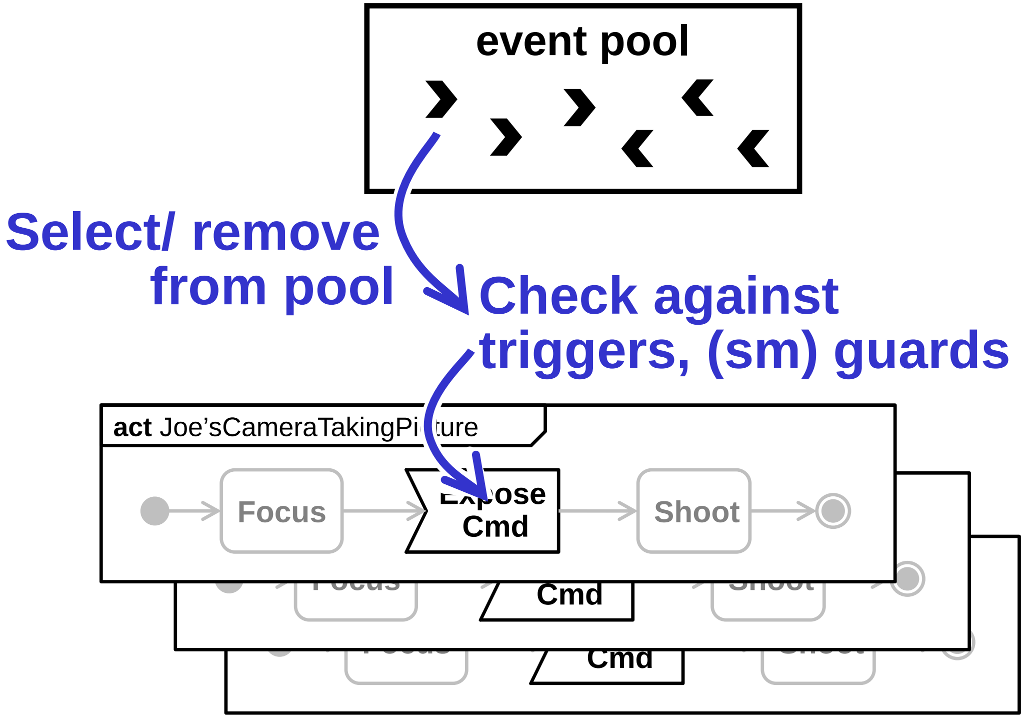
System Model (M1)



- Enable multiple behaviors to react to the same event.
- Definitely not “remove” each others’ events from the pool.
- Same for events arriving at separate ports (see PSCS/PSSM).

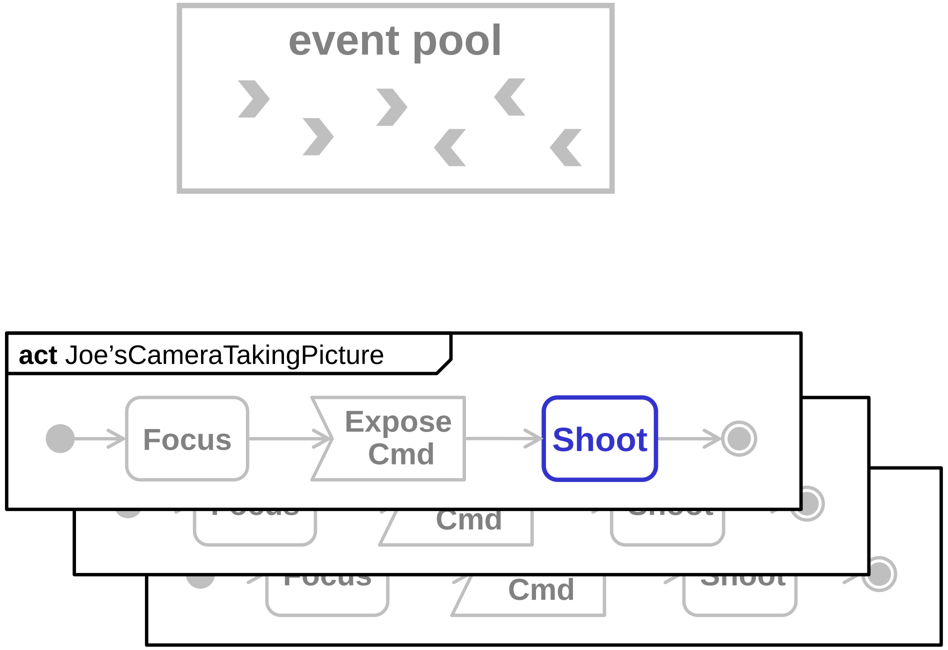
(UMLish) Run to Completion (RTC)

- Process events separately from actions.



Select/ remove from pool

Check against triggers, (sm) guards

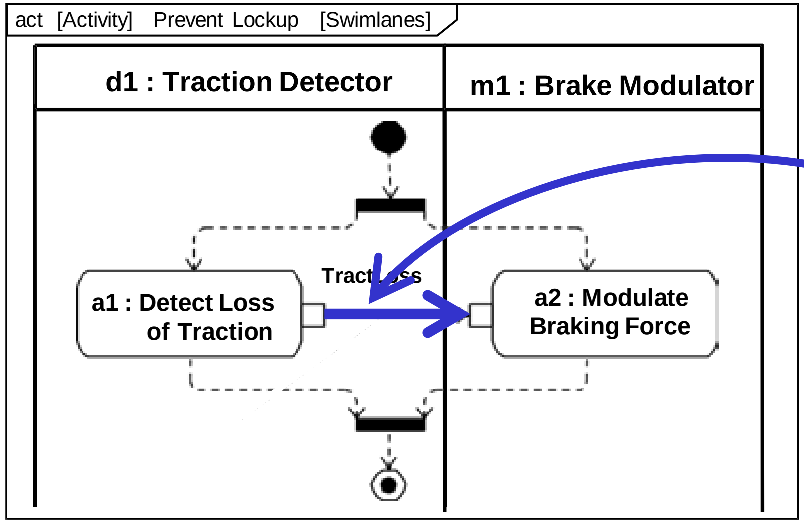


1) Process one event

2) Act (maybe)

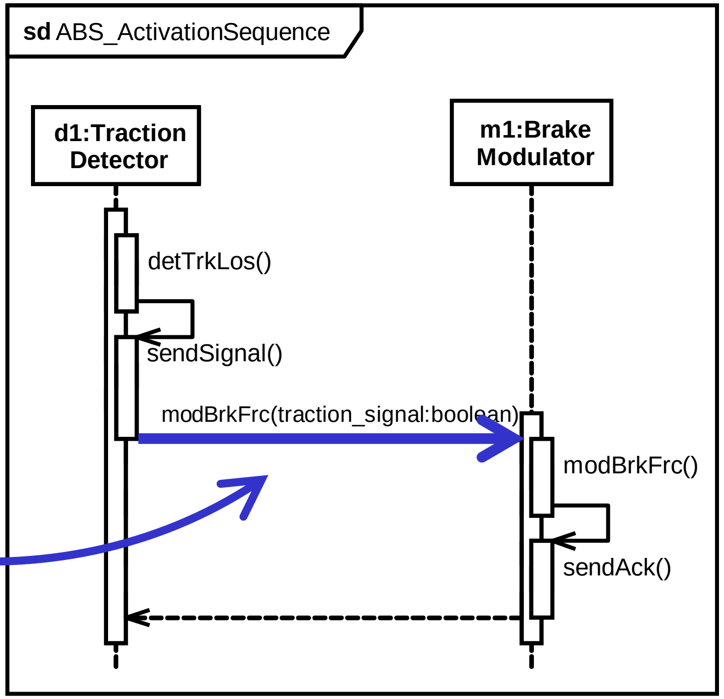
3) Repeat

UML/SysML Interactions Problem

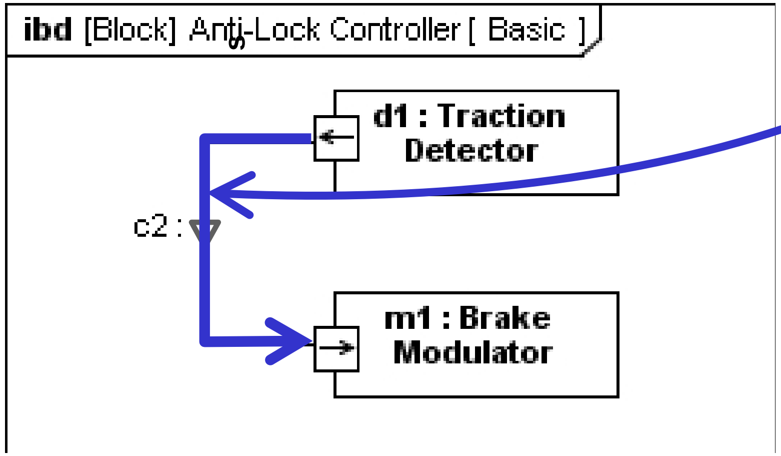

act [Activity] Prevent Lockup [Swimlanes]
d1 : Traction Detector
m1 : Brake Modulator
a1 : Detect Loss of Traction
Tract Loss
a2 : Modulate Braking Force

Activity Object Flow

Same “flow”



Interaction Messages



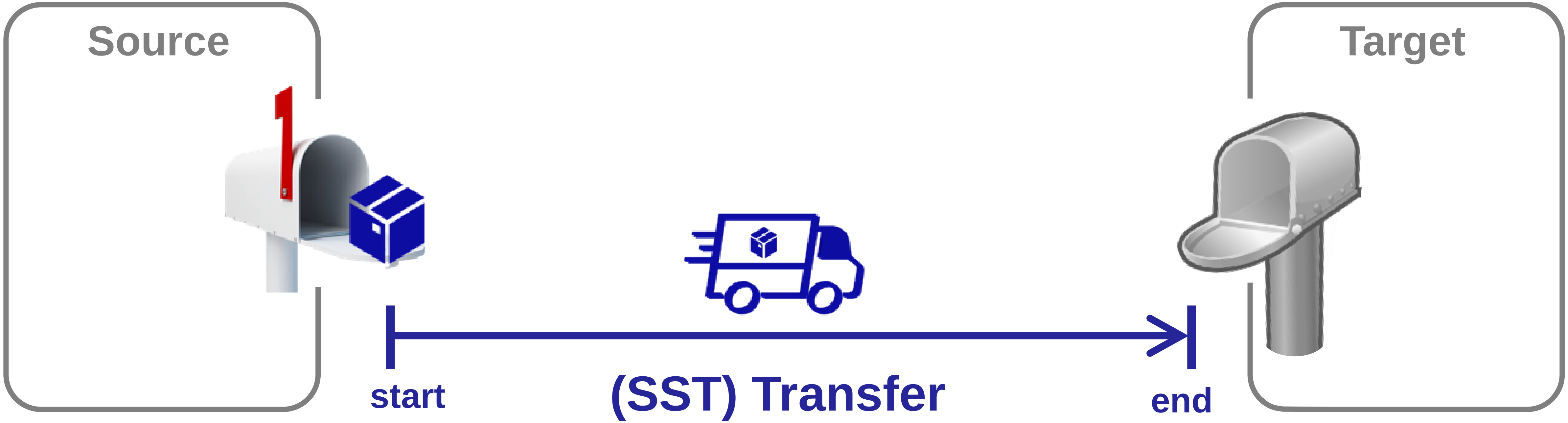
IBD Item Flows

multiple underlying models

Overview

- Event handling, requirements
- **Solutions, Kernel**
  - **Onto messages/flows**
  - Transitions
  - Accepting “events”
- Solutions, SysML
  - Accept and send actions
  - Sequence diagrams
  - Flows
- Summary

Onto Messages/Flows



- Transfers take time and space.
- Source and target are not involved in transfers
  - Except source provides the items, target receives them.

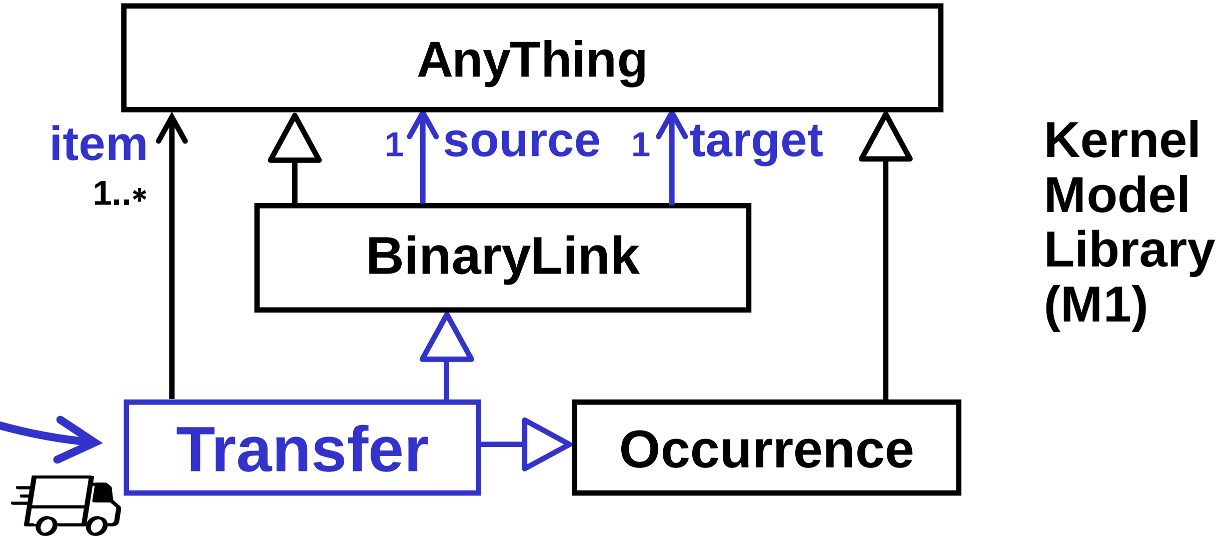
Onto Message/Flow

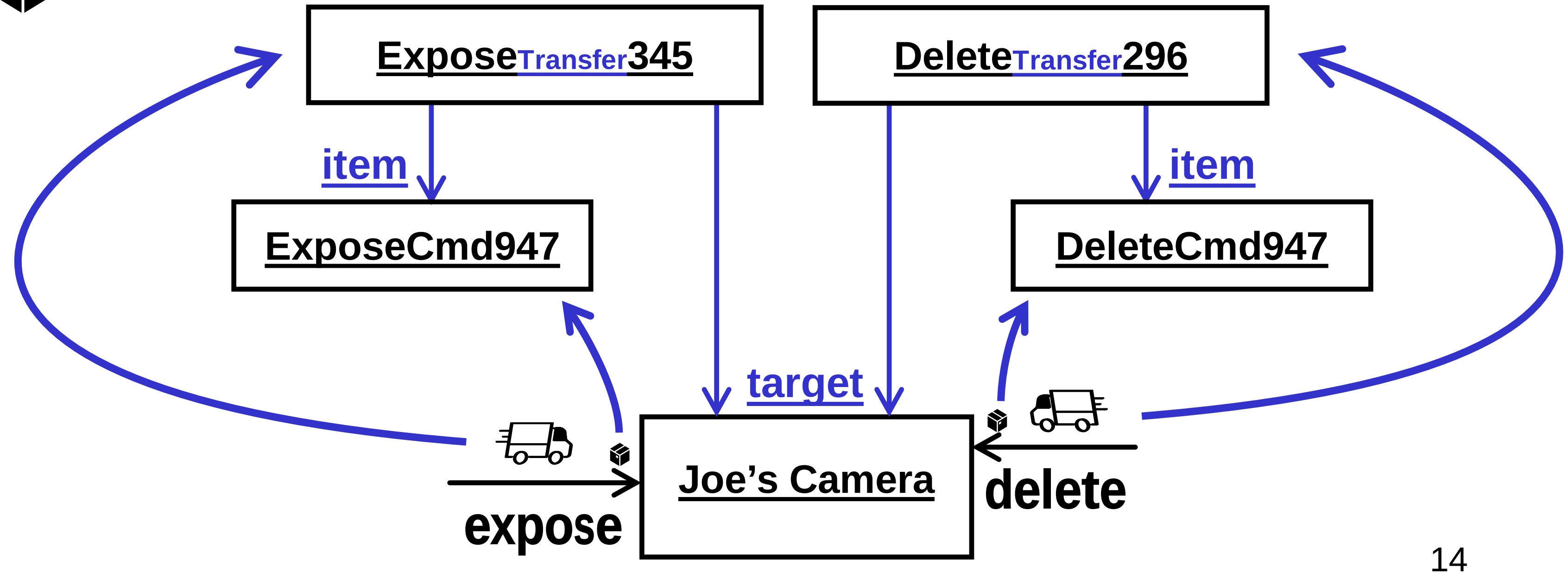
- § A kind of **occurrence**
  - – and binary link
- § **Start** when item is picked up from **source**
- § **End** when item is delivered to **target**

Real/simulation time (M0)

Outside the camera

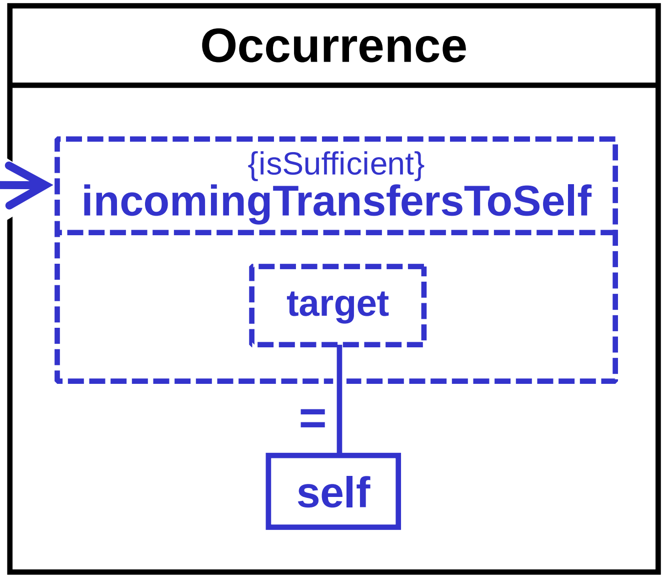
Not under camera's control





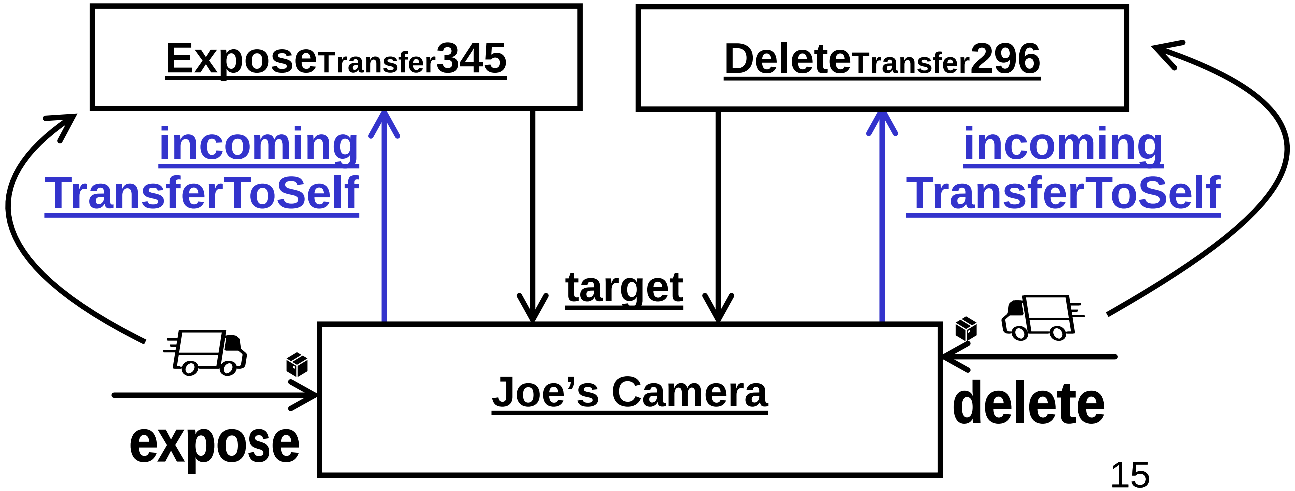
Onto Event Pools

- § Library feature …
  - – … identifying all transfers targeting each occurrence (over entire life).
  - – Only queried/matched, not modified.
- § Onto "event handling"
  - – Temporal requirements on reacting to pool contents.



Kernel Model Library (M1)

Outside the camera
Not under camera's control

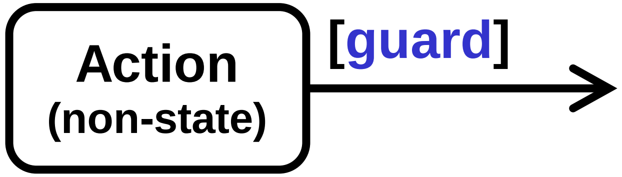


Real/simulation time (M0)

Overview

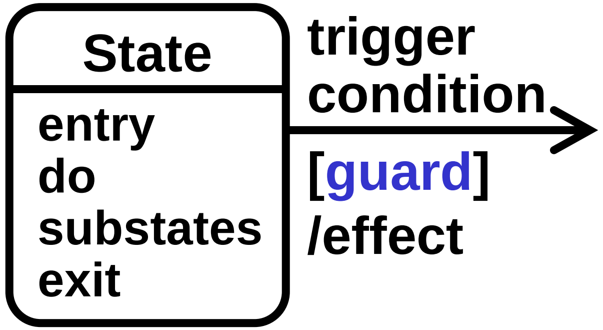
- Event handling, requirements
- Solutions, Kernel
  - Onto messages/flows
  - Transitions
  - Accepting “events”
- Solutions, SysML
  - Accept and send actions
  - Sequence diagrams
  - Flows
- Summary

“Event Processing”: Transitions



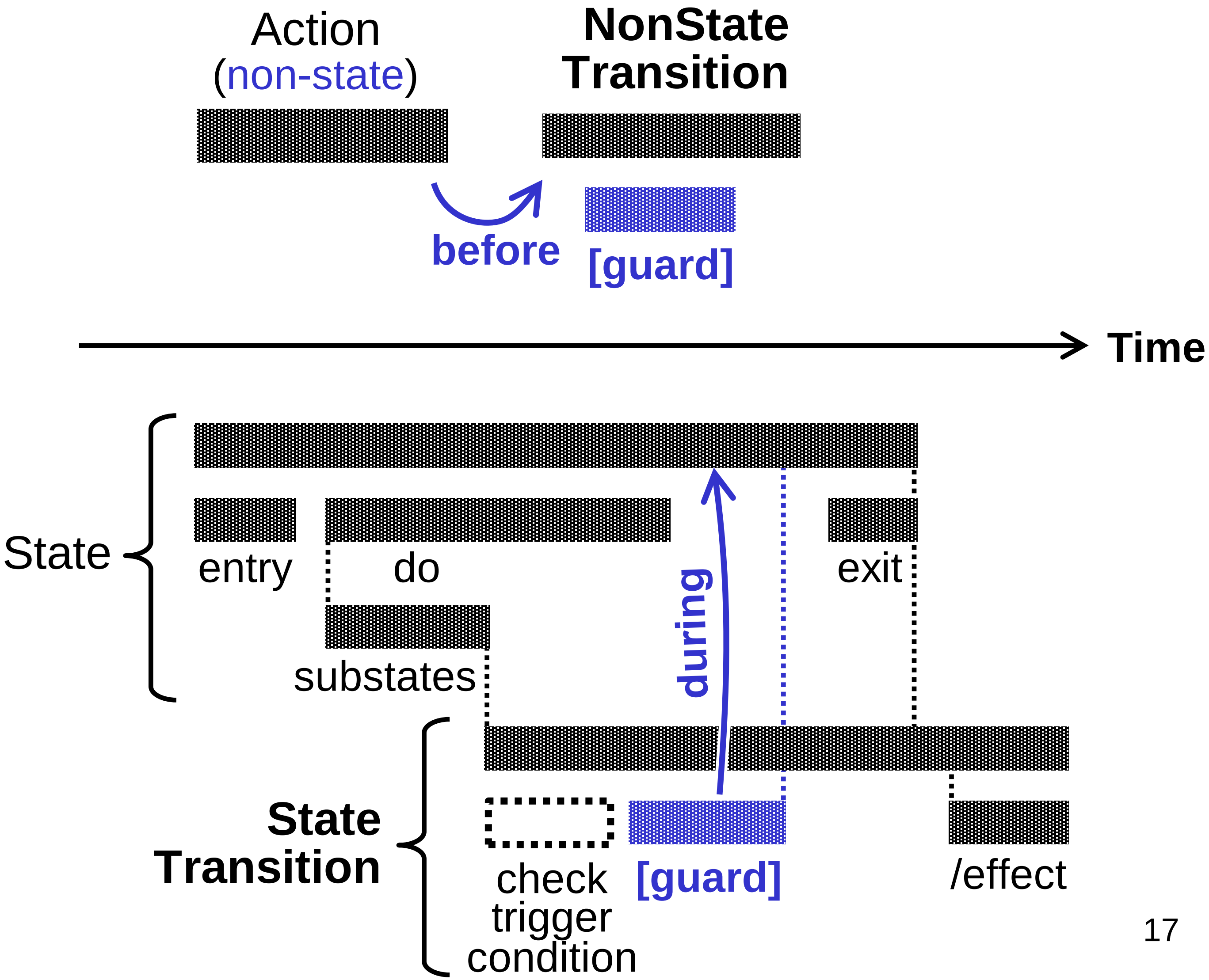
Action
(non-state)
[guard]

Transitions happen after
non-state actions



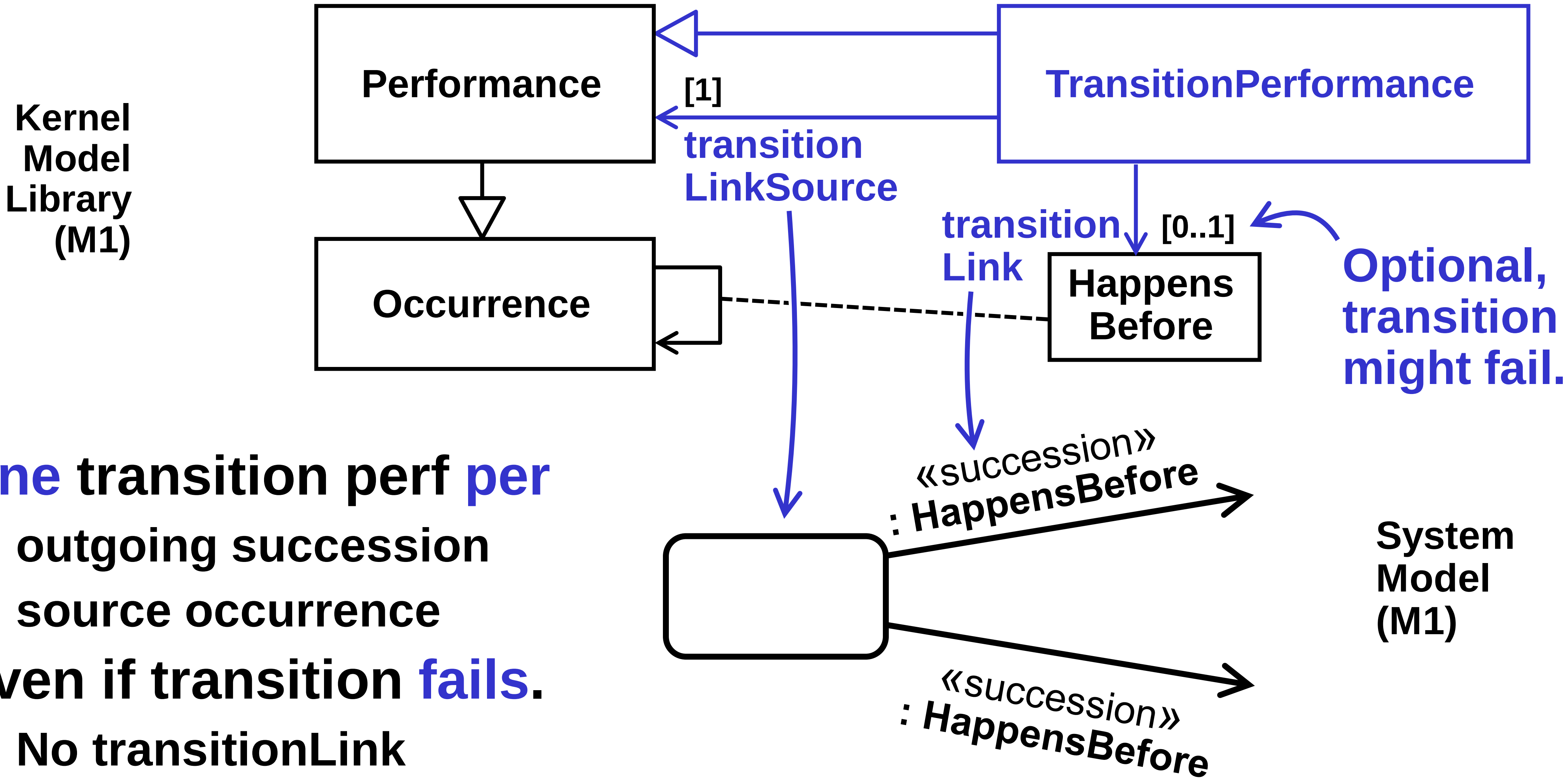
State
entry
do
substates
exit
trigger
condition
[guard]
/effect

Transitions overlap states.
Guards evaluated during states.



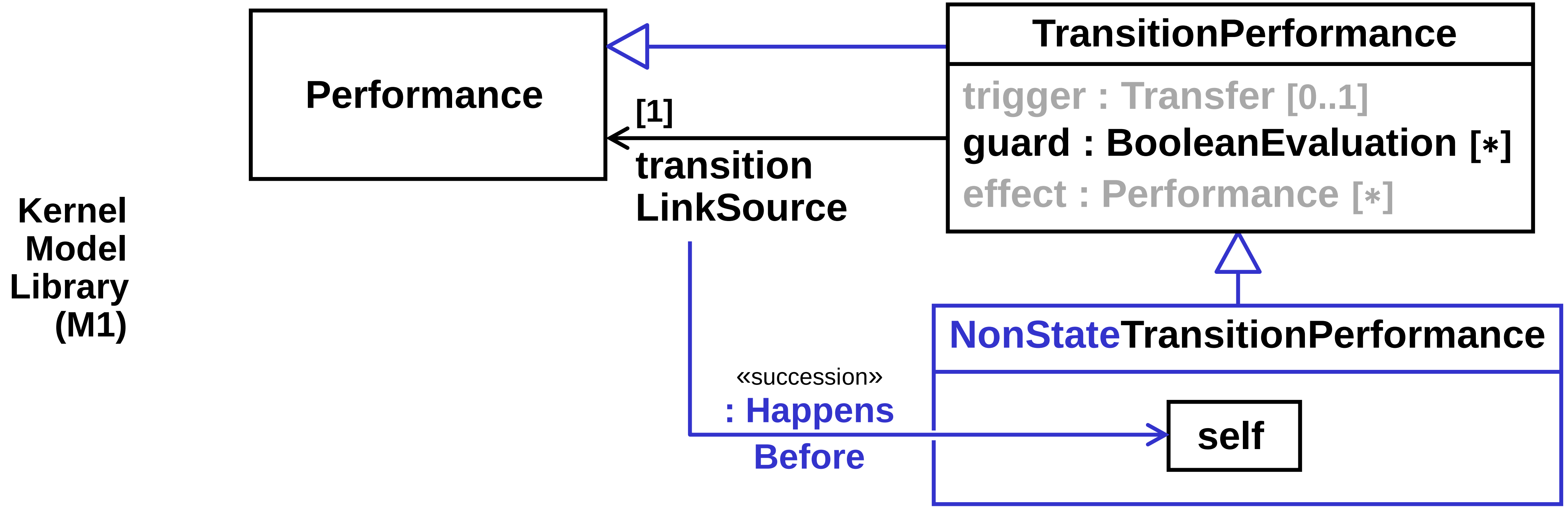
Action
(non-state)
NonState
Transition
before
[guard]
Time
State
entry
do
substates
exit
during
State
Transition
check
trigger
condition
[guard]
/effect

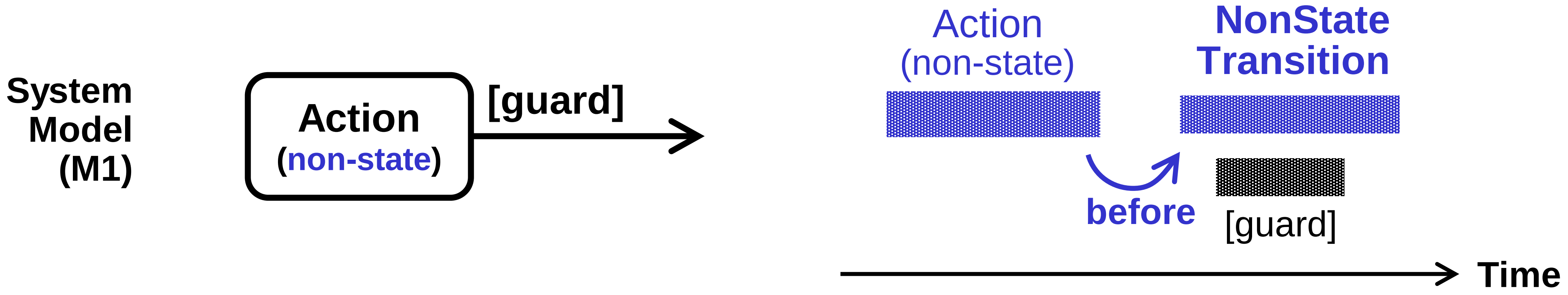
SST Transition Performances



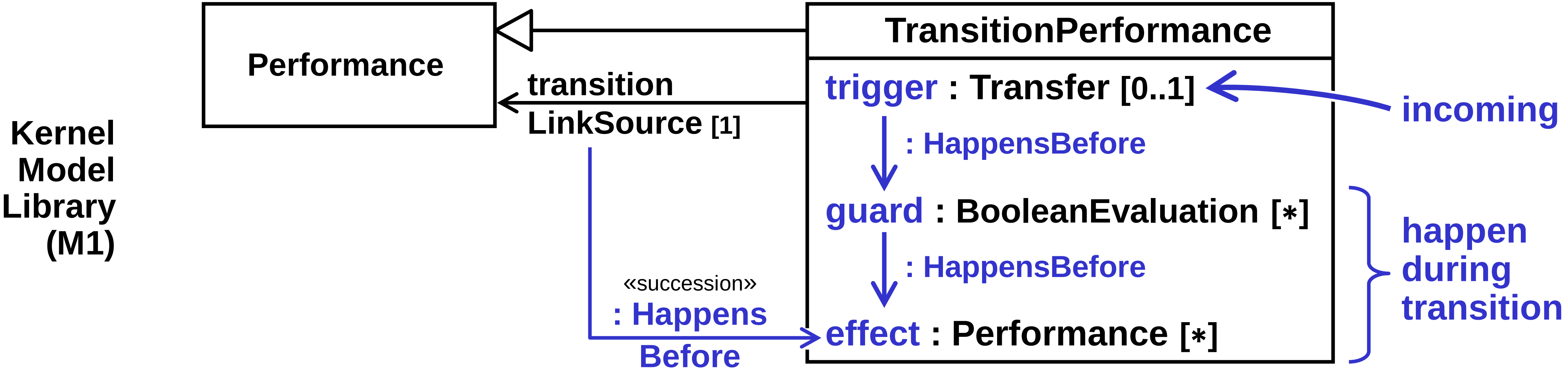
- One transition perf per
  - outgoing succession
  - source occurrence
- Even if transition fails.
  - No transitionLink

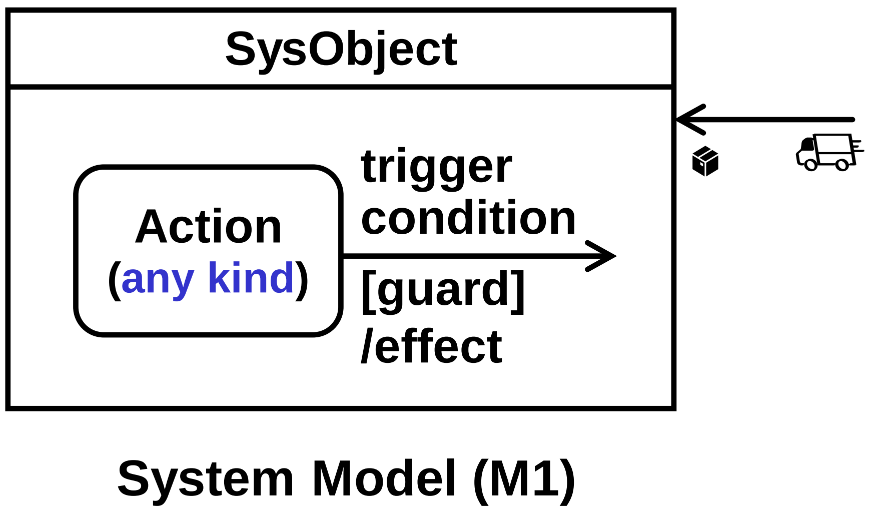
NonState Transition Performances, Timing, 1

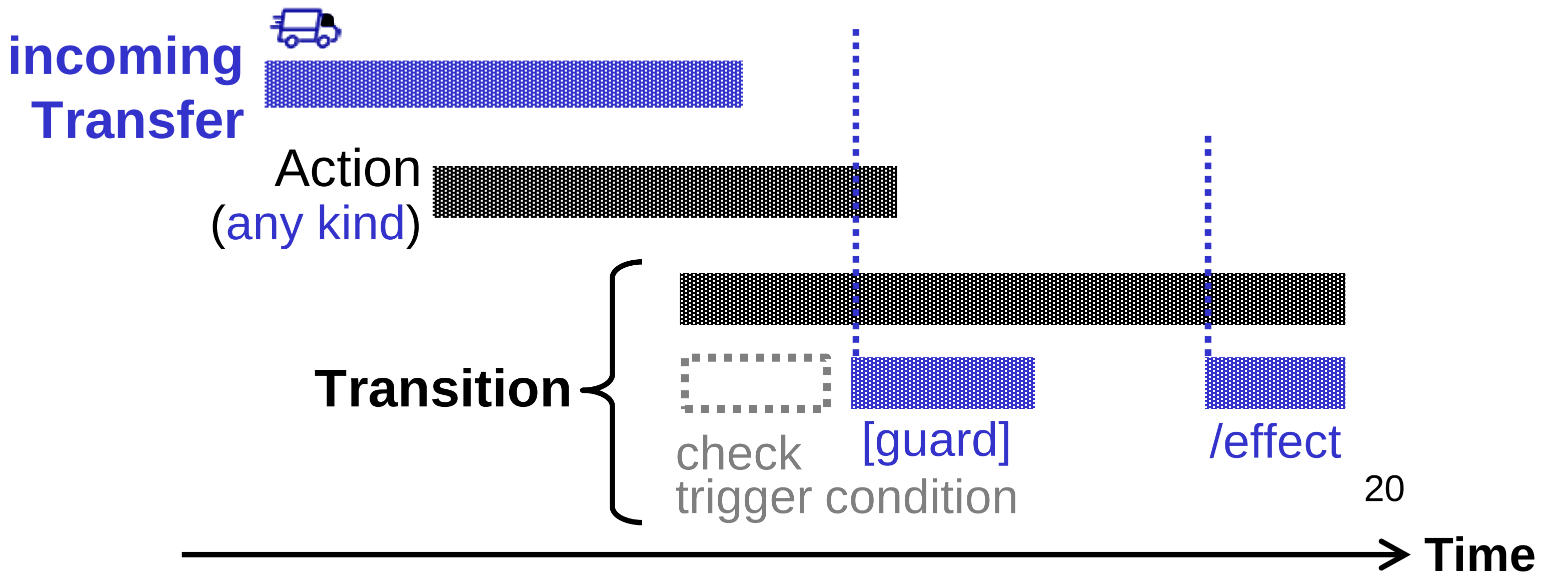




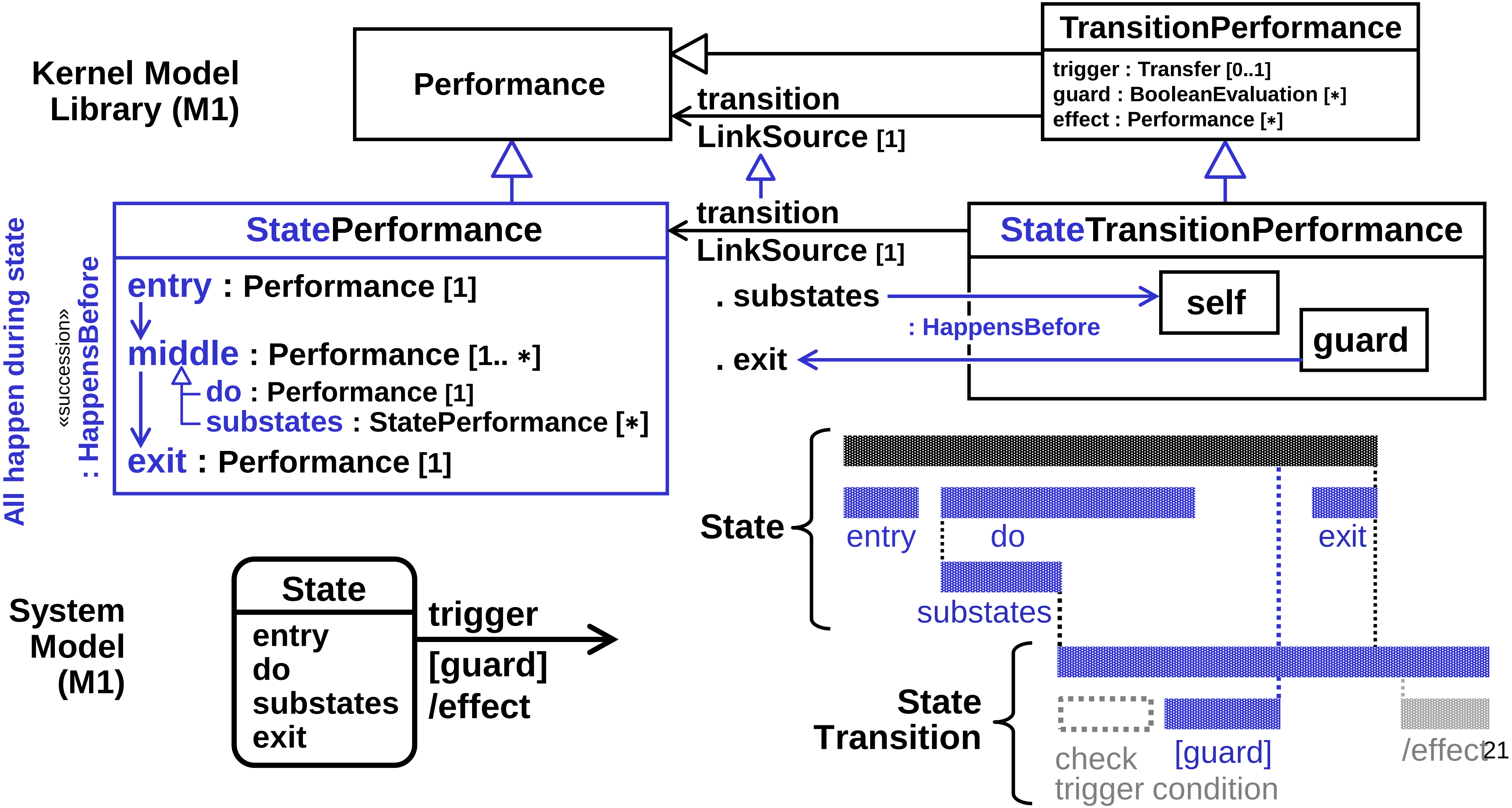
Transition Performances, Timing, 2







State (Transition) Performances, Timing, 3



Kernel Model Library (M1)

Performance

TransitionPerformance

trigger : Transfer [0..1]
guard : BooleanEvaluation [*]
effect : Performance [*]

transition
LinkSource [1]

transition
LinkSource [1]

StatePerformance

entry : Performance [1]
middle : Performance [1.. *]
do : Performance [1]
substates : StatePerformance [*]
exit : Performance [1]

All happen during state

«succession»
: HappensBefore

StateTransitionPerformance

. substates
: HappensBefore
. exit

self
guard

State

entry
do
exit
substates

State Transition

check
trigger condition
[guard]
/effect

System Model (M1)

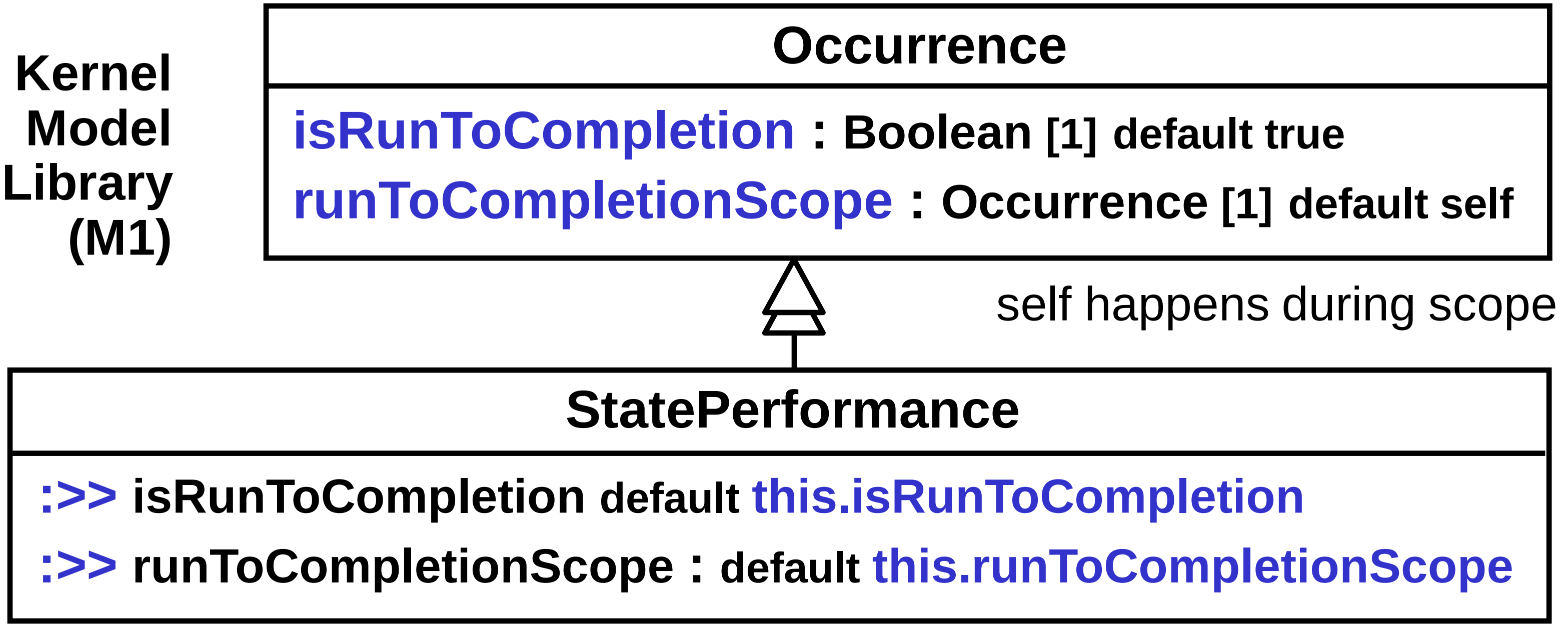
State
entry
do
substates
exit

trigger
[guard]
/effect

Run To Completion

- No Transitions (within scope) during entry.
  - No guard evals (within scope) during exit, effect

Kernel Model Library (M1)


Occurrence

isRunToCompletion : Boolean [1] default true

runToCompletionScope : Occurrence [1] default self

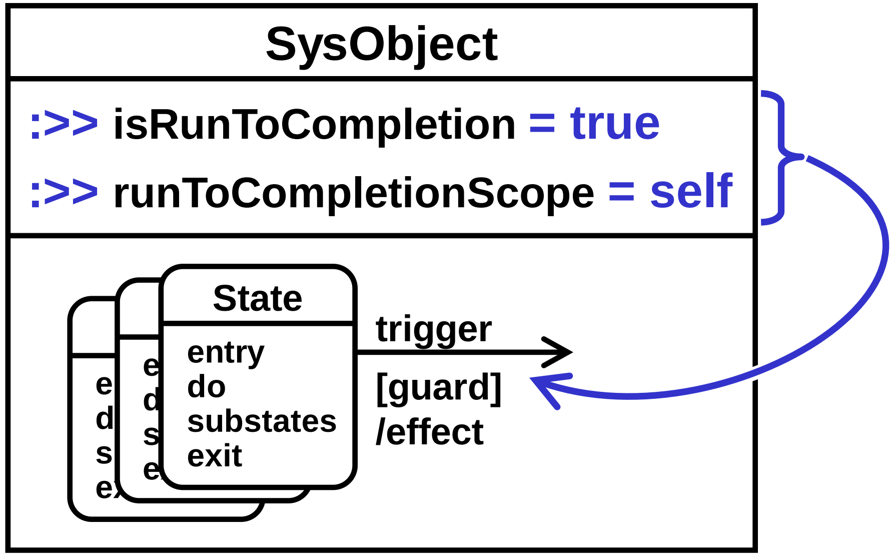
self happens during scope

StatePerformance

:>> isRunToCompletion default this.isRunToCompletion

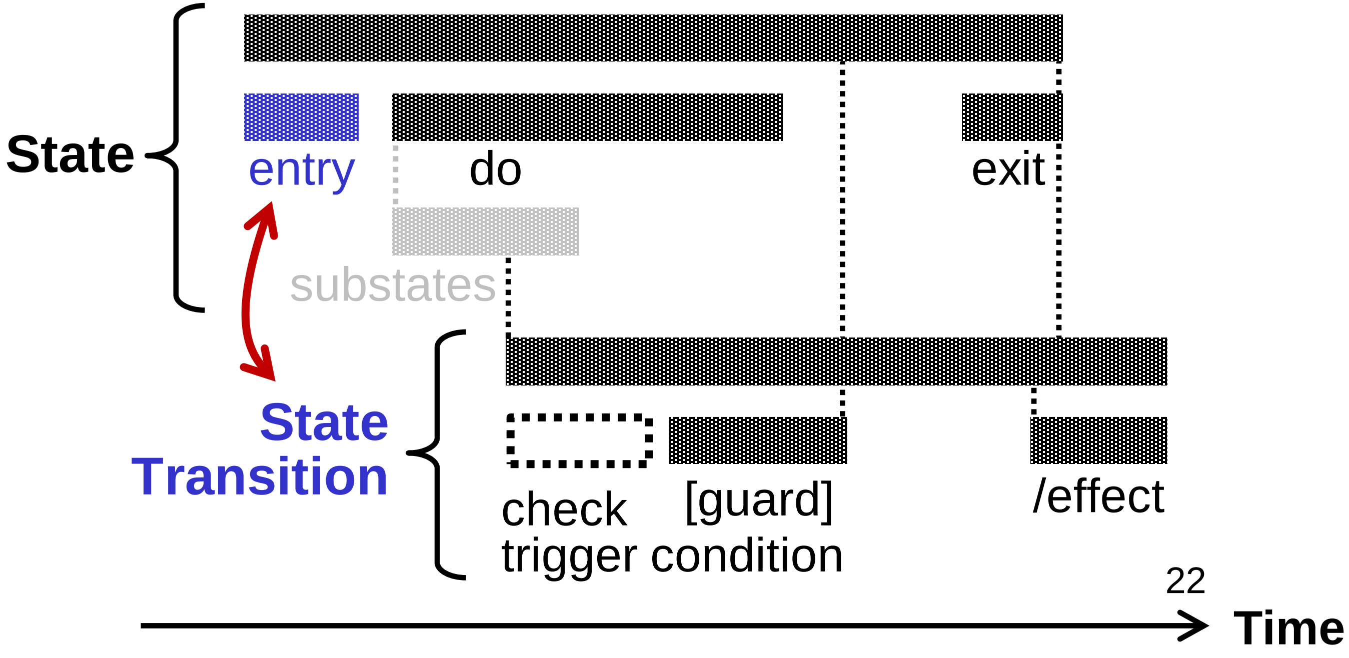
:>> runToCompletionScope : default this.runToCompletionScope

System Model (M1)


SysObject

:>> isRunToCompletion = true

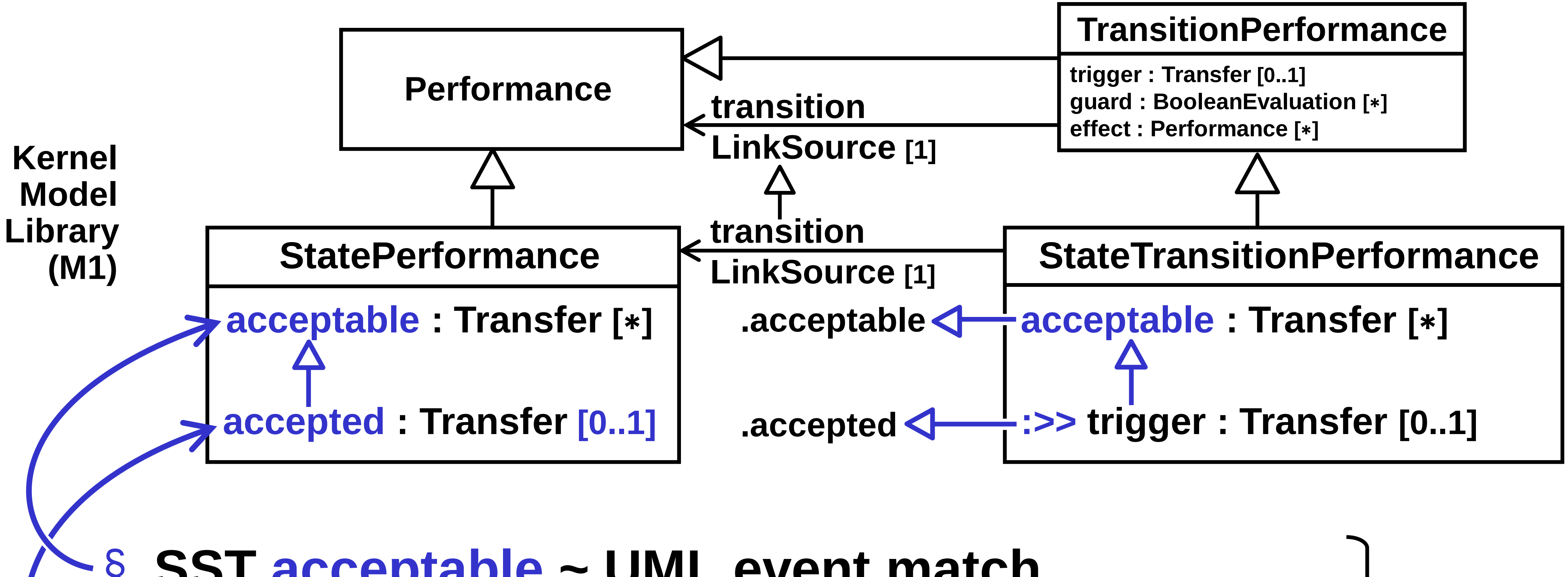
:>> runToCompletionScope = self



Overview

- Event handling, requirements
- Solutions, Kernel
  - Onto messages/flows
  - Transitions
  - **Accepting “events”**
- Solutions, SysML
  - Accept and send actions
  - Sequence diagrams
  - Flows
- Summary

State (Transition) Performances, Timing, 4

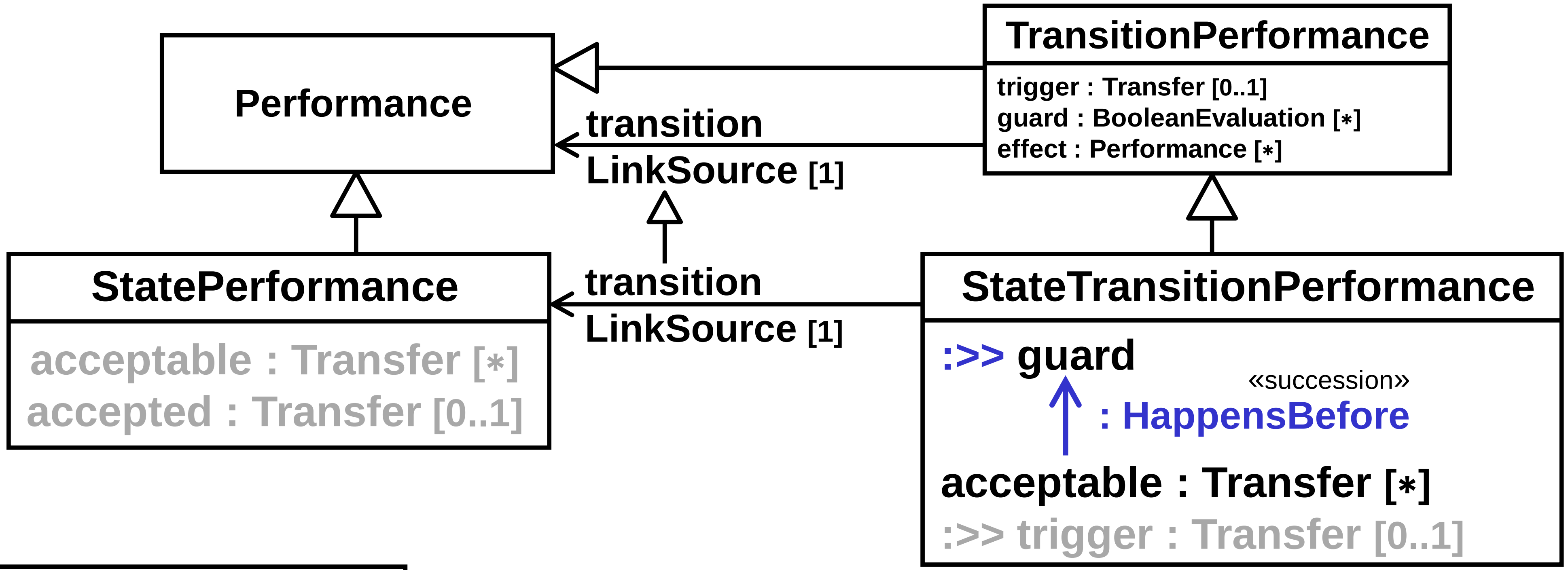

Kernel Model Library (M1)
Performance
TransitionPerformance
trigger : Transfer [0..1]
guard : BooleanEvaluation [*]
effect : Performance [*]
transition LinkSource [1]
transition LinkSource [1]
StatePerformance
acceptable : Transfer [*]
accepted : Transfer [0..1]
StateTransitionPerformance
acceptable : Transfer [*]
:>> trigger : Transfer [0..1]
.acceptable
.accepted

- SST acceptable ~ UML event match
  - (not UML enabled, which includes true guards)
- SST accepted/trigger ~ UML “fired”

} conditions specified on usage

State (Transition) Performances, Timing, 5

Kernel
Model
Library
(M1)



Performance

TransitionPerformance

trigger : Transfer [0..1]
guard : BooleanEvaluation [*]
effect : Performance [*]

transition
LinkSource [1]

StatePerformance

acceptable : Transfer [*]
accepted : Transfer [0..1]

transition
LinkSource [1]

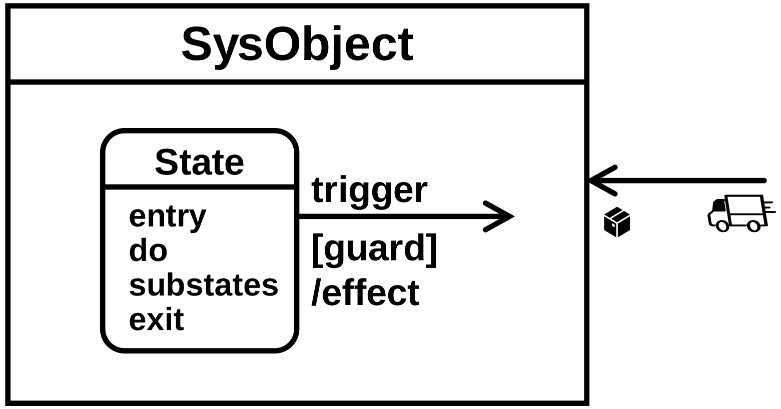
StateTransitionPerformance

:>> guard

«succession»
: HappensBefore

acceptable : Transfer [*]
:>> trigger : Transfer [0..1]

System
Model
(M1)

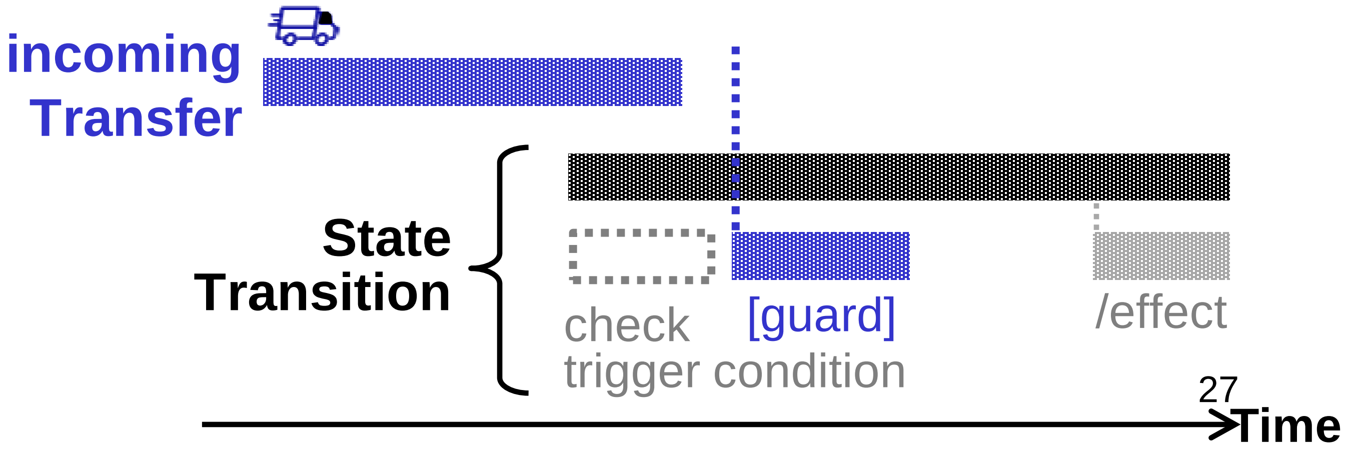


SysObject

State

entry
do
substates
exit

trigger
[guard]
/effect



incoming
Transfer

State
Transition

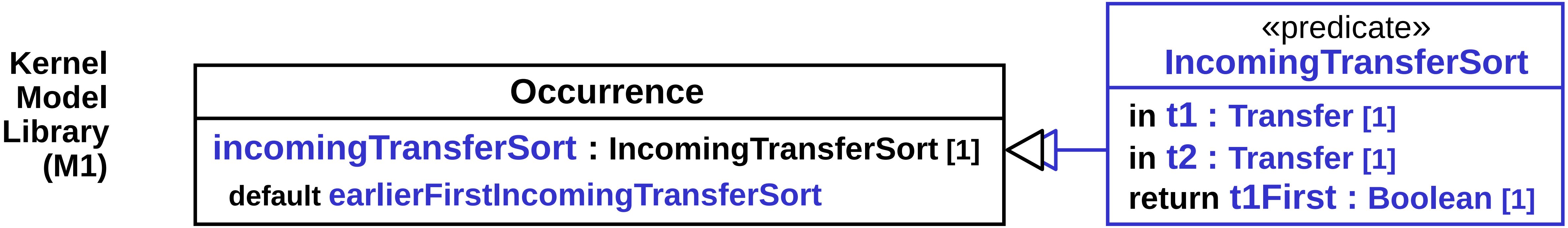
check
trigger condition

[guard]

/effect

Time

Prioritizing Incoming Transfers



Kernel Model Library (M1)

Occurrence

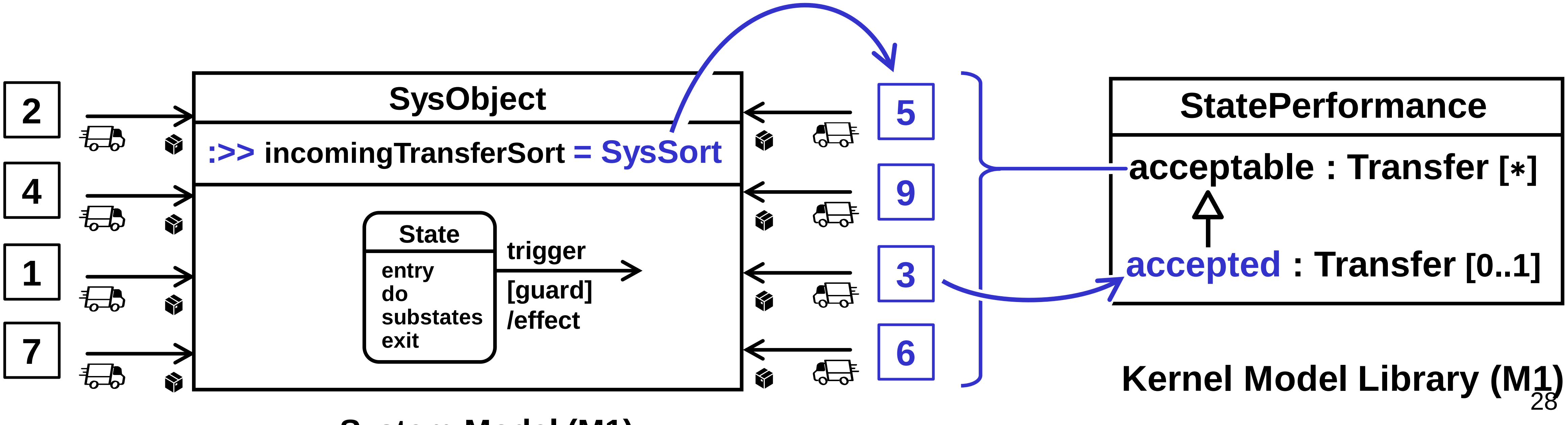
incomingTransferSort : IncomingTransferSort [1]

default earlierFirstIncomingTransferSort

«predicate»
IncomingTransferSort

in t1 : Transfer [1]
in t2 : Transfer [1]
return t1First : Boolean [1]

```
feature earlierFirstIncomingTransferSort : IncomingTransferSort {
  inv { t1First == includes(t1.endShot.successors, t2.endShot) } }
```



2
4
1
7
SysObject
:>> incomingTransferSort = SysSort
State
entry
do
substates
exit
trigger
[guard]
/effect
5
9
3
6
StatePerformance
acceptable : Transfer [*]
accepted : Transfer [0..1]

System Model (M1)

Kernel Model Library (M1)

Dispatch

- Not acceptable if accepted before
  - or higher in sort order than last accepted.

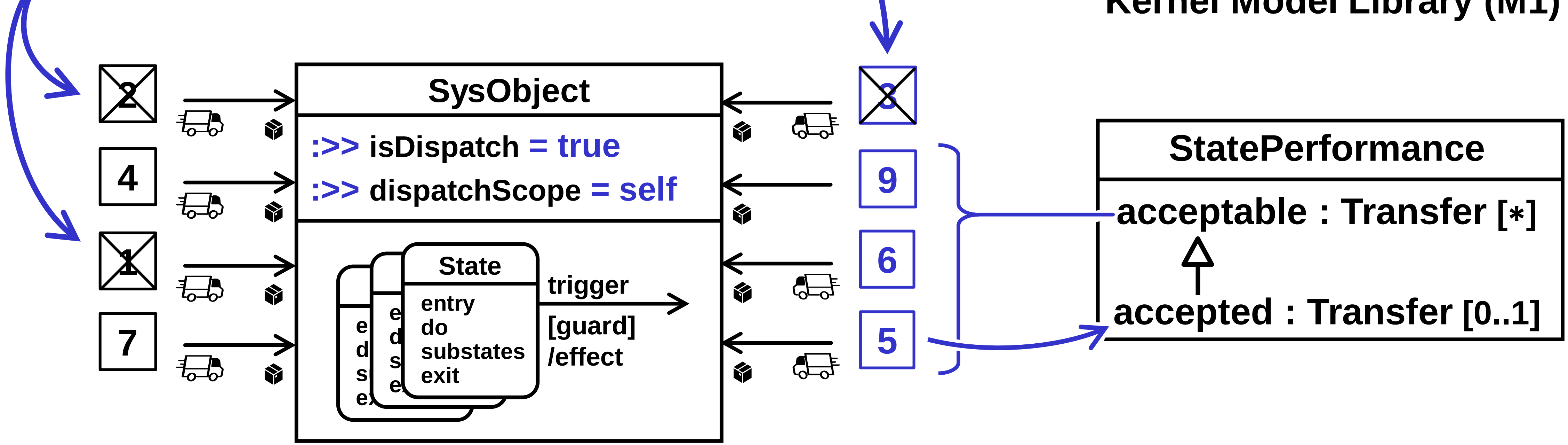
Occurrence

isDispatch : Boolean [1] default false

dispatchScope : Occurrence [1] default self

self happens during scope

Kernel Model Library (M1)



2

4

1

7

SysObject

:>> isDispatch = true

:>> dispatchScope = self

State

entry
do
substates
exit

trigger
[guard]
/effect

3

9

6

5

StatePerformance

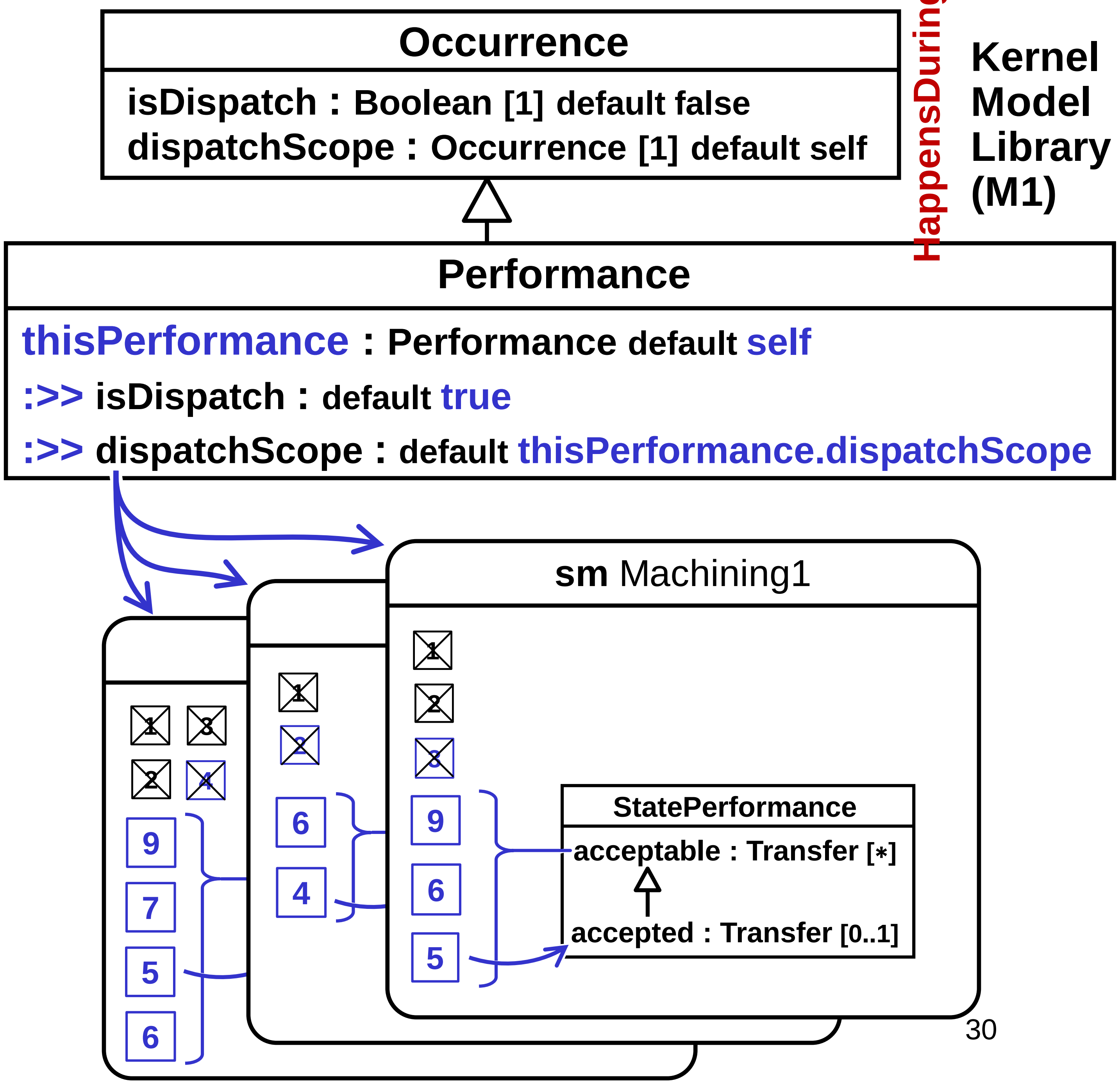
acceptable : Transfer [*]

accepted : Transfer [0..1]

System Model (M1)

Dispatch per (top) Performance

- each with its own dispatching, prioritization



Kernel Model Library (M1)

HappensDuring

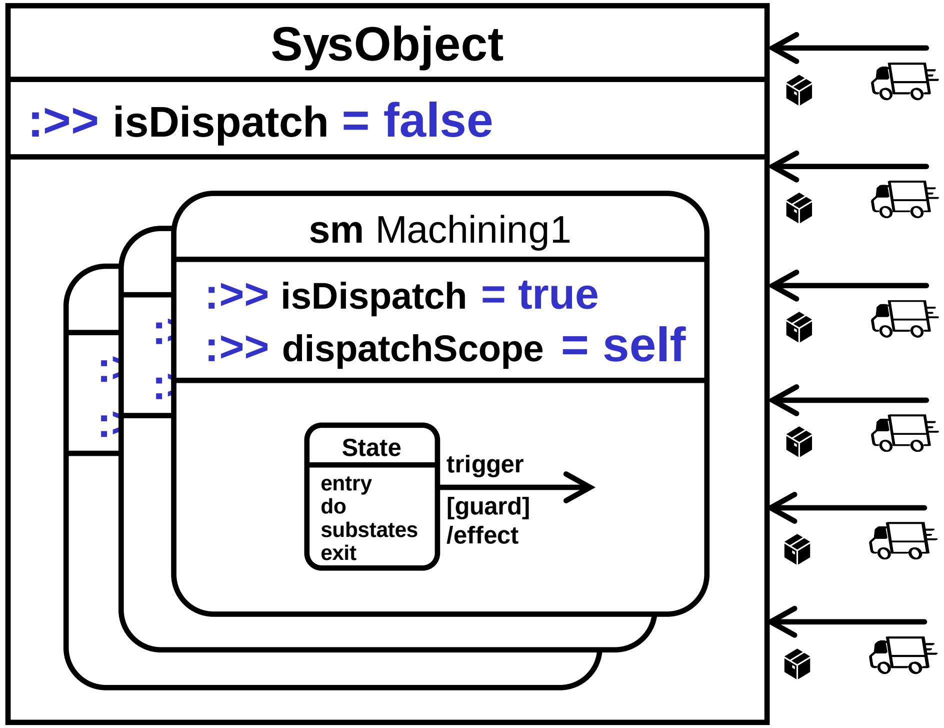
Occurrence

isDispatch : Boolean [1] default false
dispatchScope : Occurrence [1] default self

Performance

thisPerformance : Performance default self
:>> isDispatch : default true
:>> dispatchScope : default thisPerformance.dispatchScope

System Model (M1)



SysObject

:>> isDispatch = false

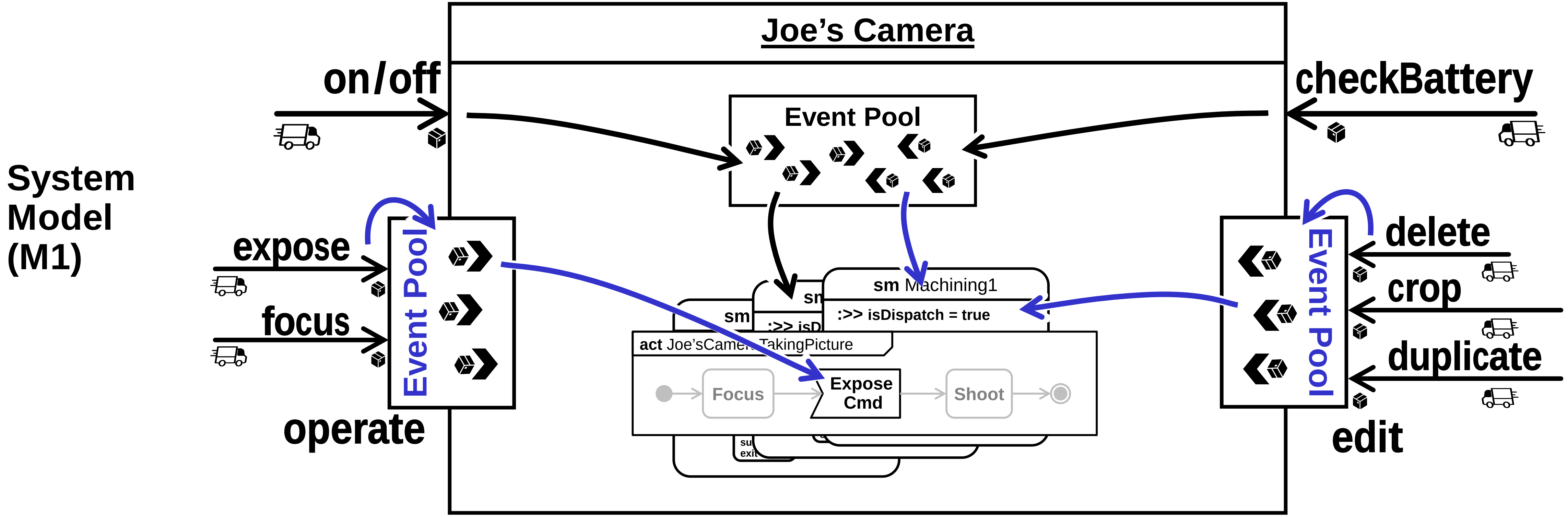
sm Machining1

:>> isDispatch = true
:>> dispatchScope = self

Overview

- Event handling, requirements
- Solutions, Kernel
  - Onto messages/flows
  - Transitions
  - Accepting “events”
- **Solutions, SysML**
  - **Accept and send actions**
  - Sequence diagrams
  - Flows
- Summary

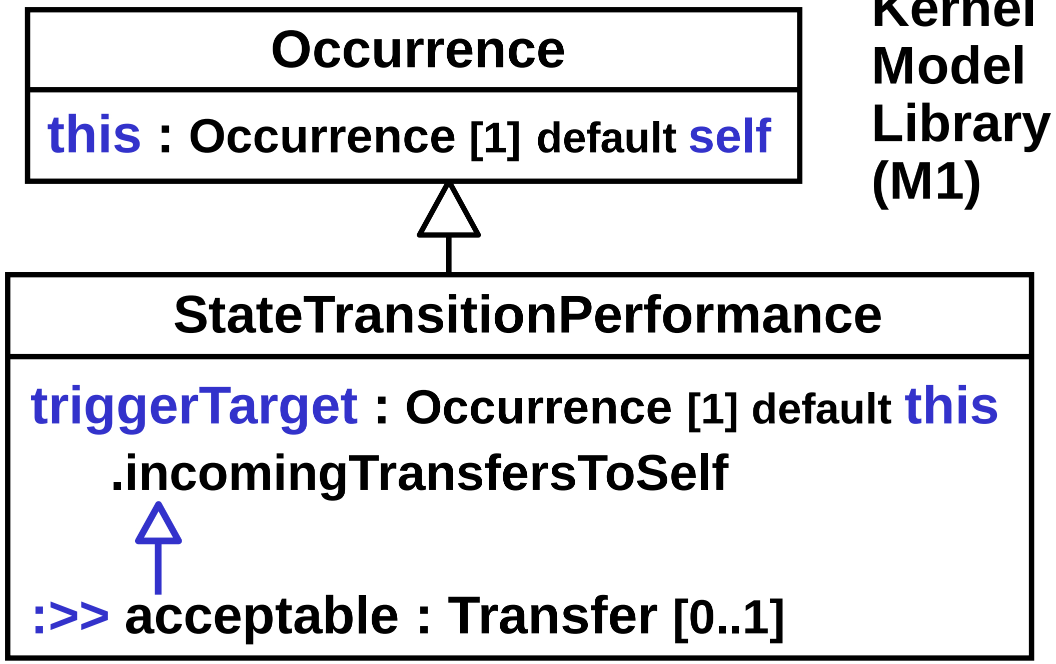
Multiple Targets, Event Pools



- Transfers can target ports …
  - … giving ports their own event pools when ports are not internally bound (incomingTransfersToSelf).

Kernel Incoming Transfer Target

- accept "via" object
  - identified by transition
  - default to "most inner" object.



Occurrence

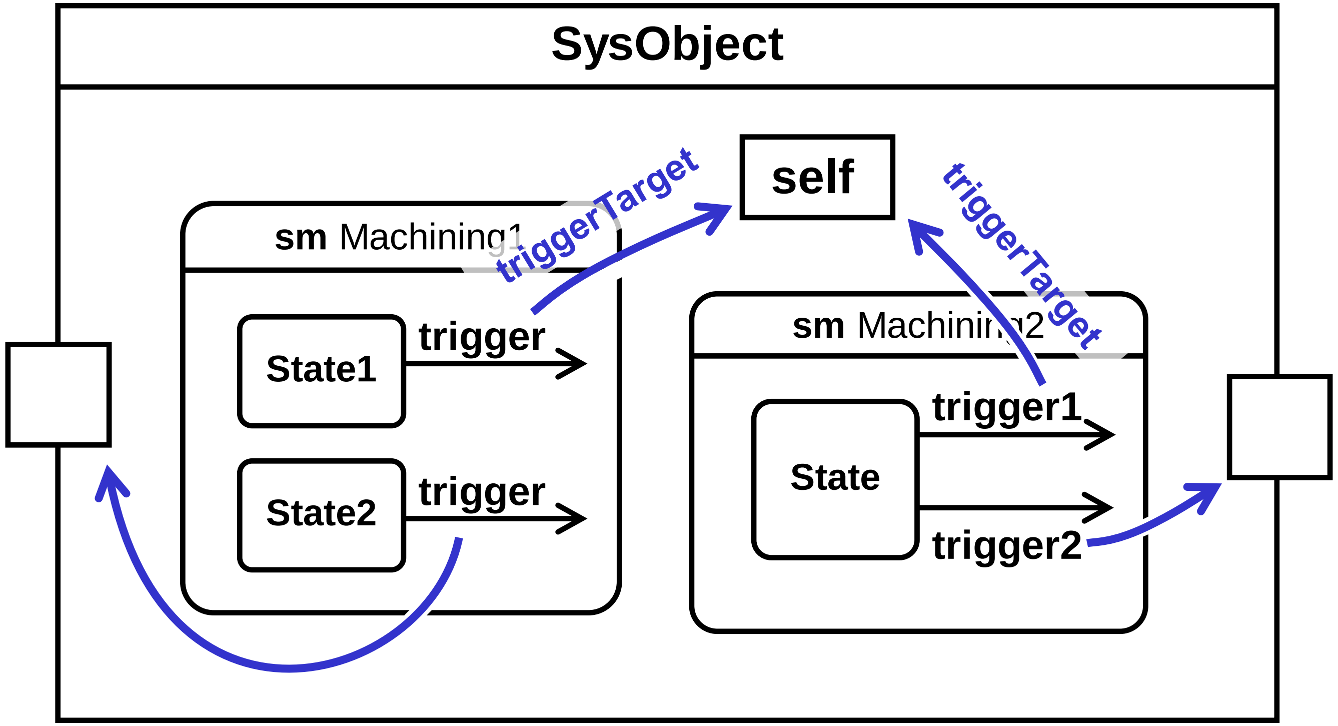
this : Occurrence [1] default self

StateTransitionPerformance

triggerTarget : Occurrence [1] default this
.incomingTransfersToSelf

:>> acceptable : Transfer [0..1]

Kernel
Model
Library
(M1)



SysObject

self

sm Machining1

State1

trigger

State2

trigger

triggerTarget

sm Machining2

State

trigger1

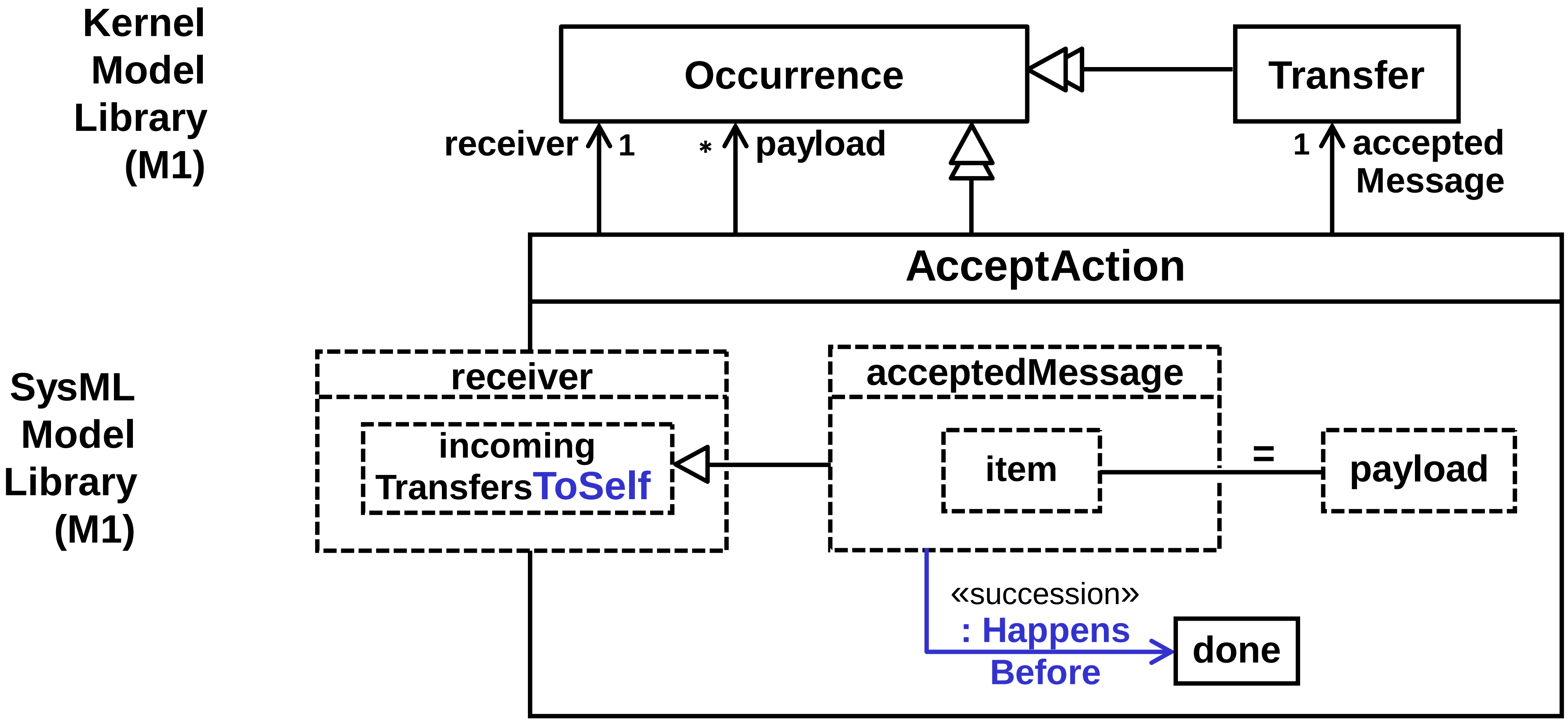
trigger2

triggerTarget

System
Model
(M1)

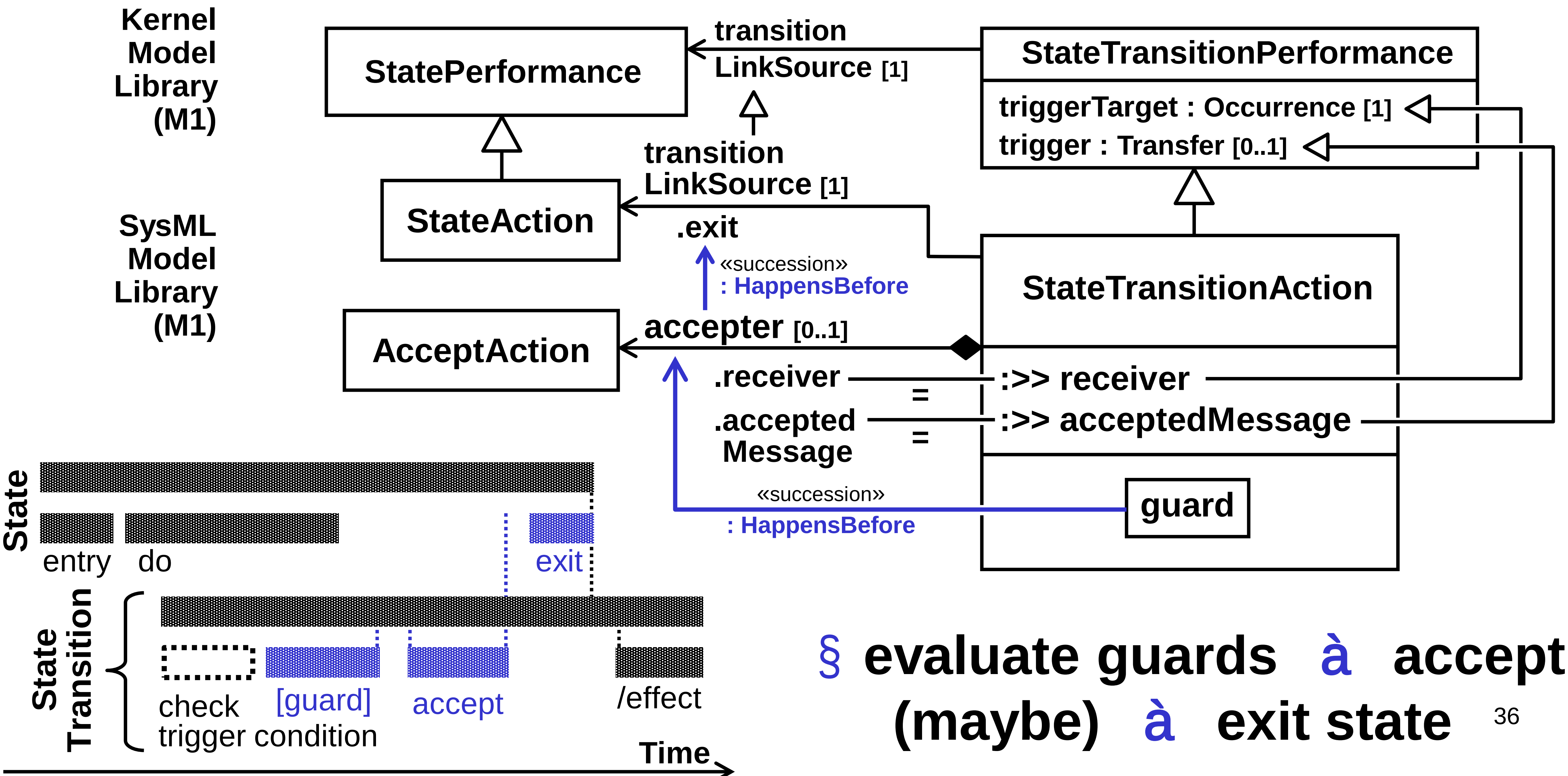
Targets can vary within objects, behaviors, even actions.

SysML Accept Actions



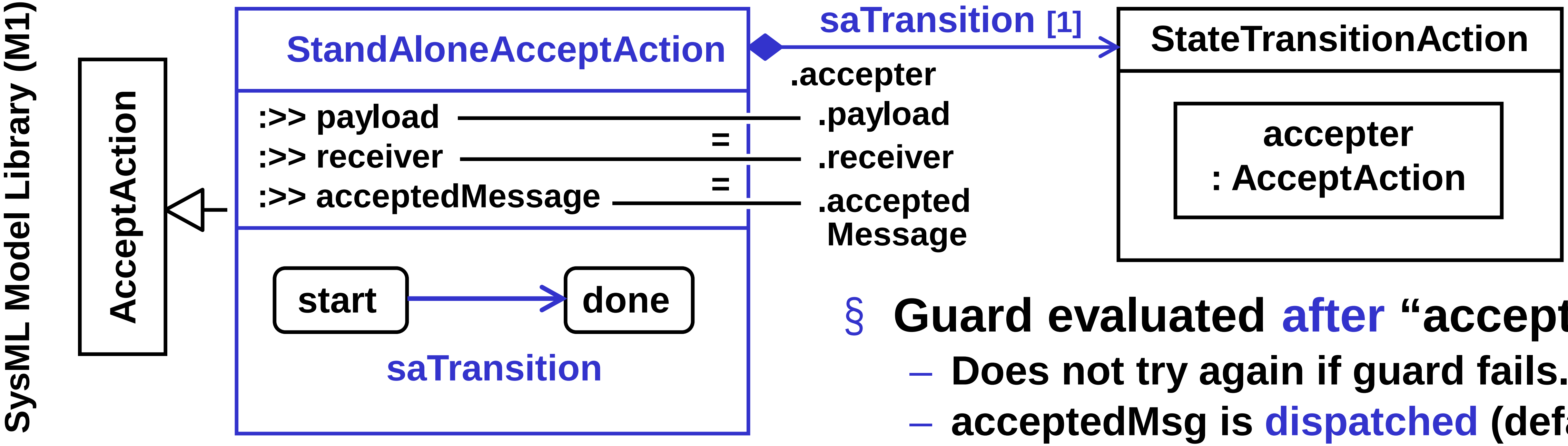
§ = Message to receiver required to match usage specs.

SysML State Transition Accept

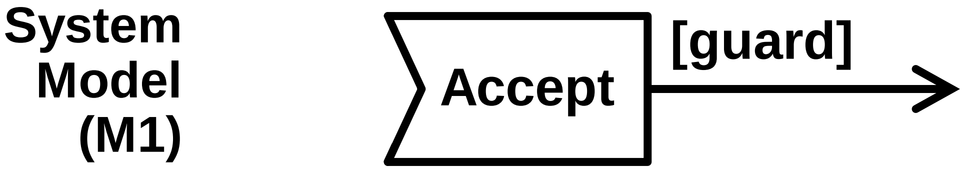


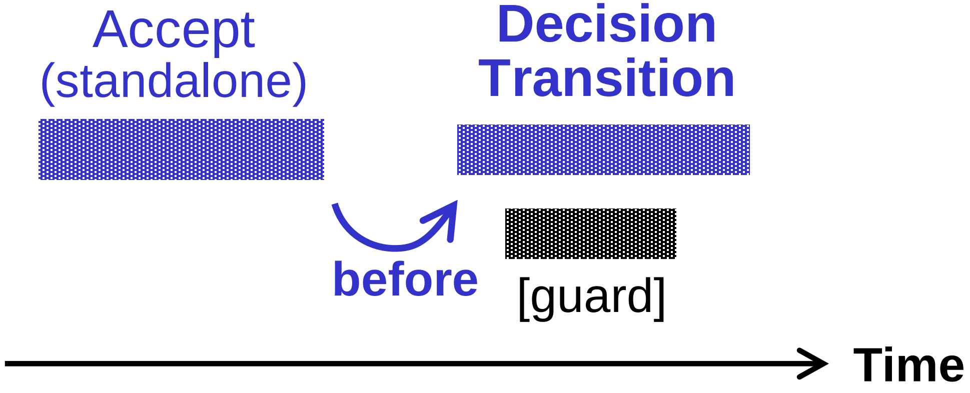
- evaluate guards à accept (maybe) à exit state

“Stand Alone” Accept

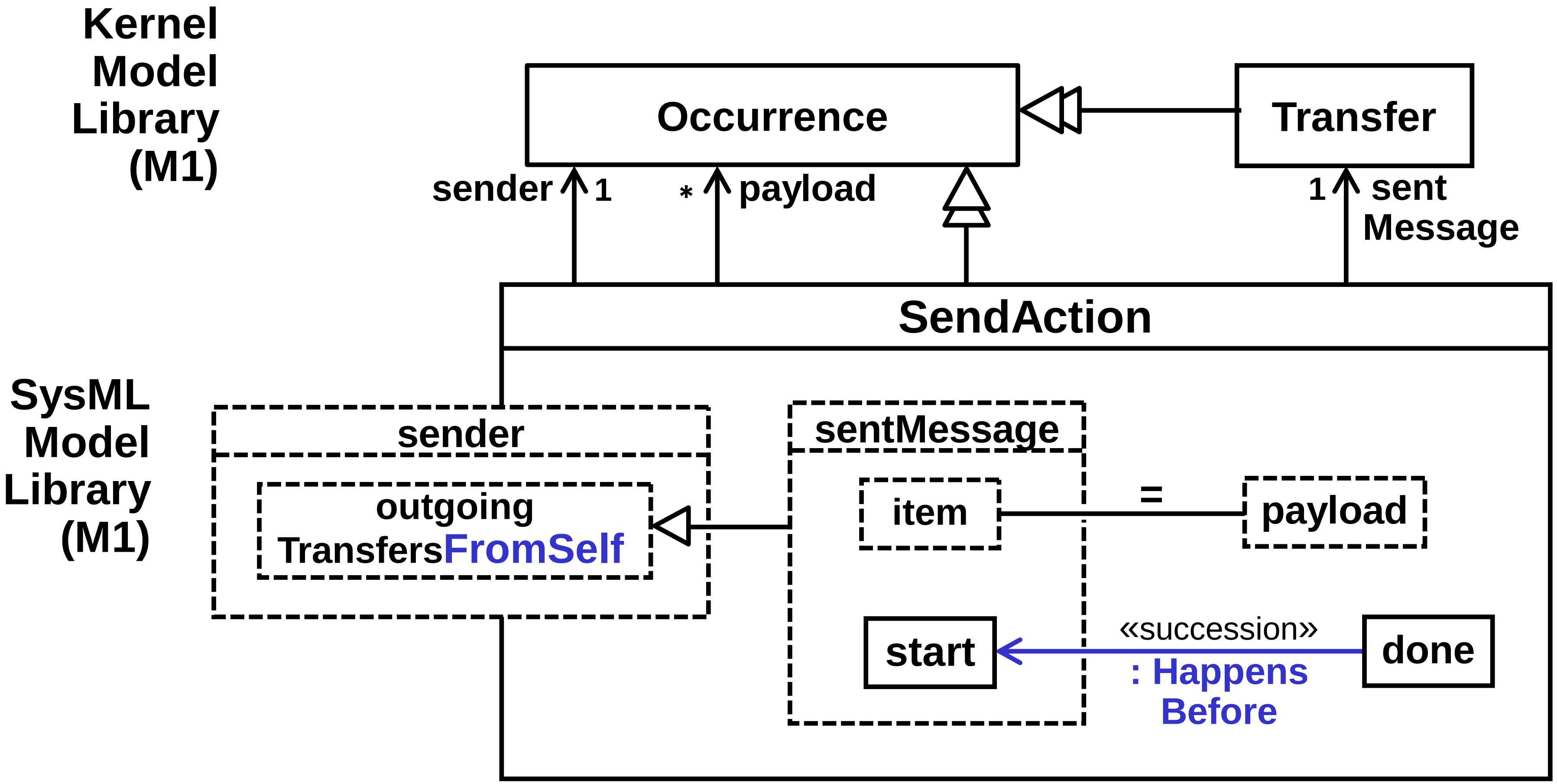


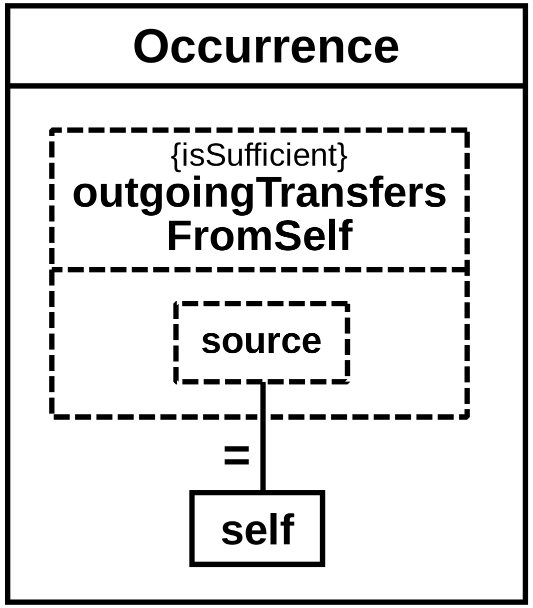
- Guard evaluated after “accept”
  - Does not try again if guard fails.
  - acceptedMsg is dispatched (default).





SysML Send Actions



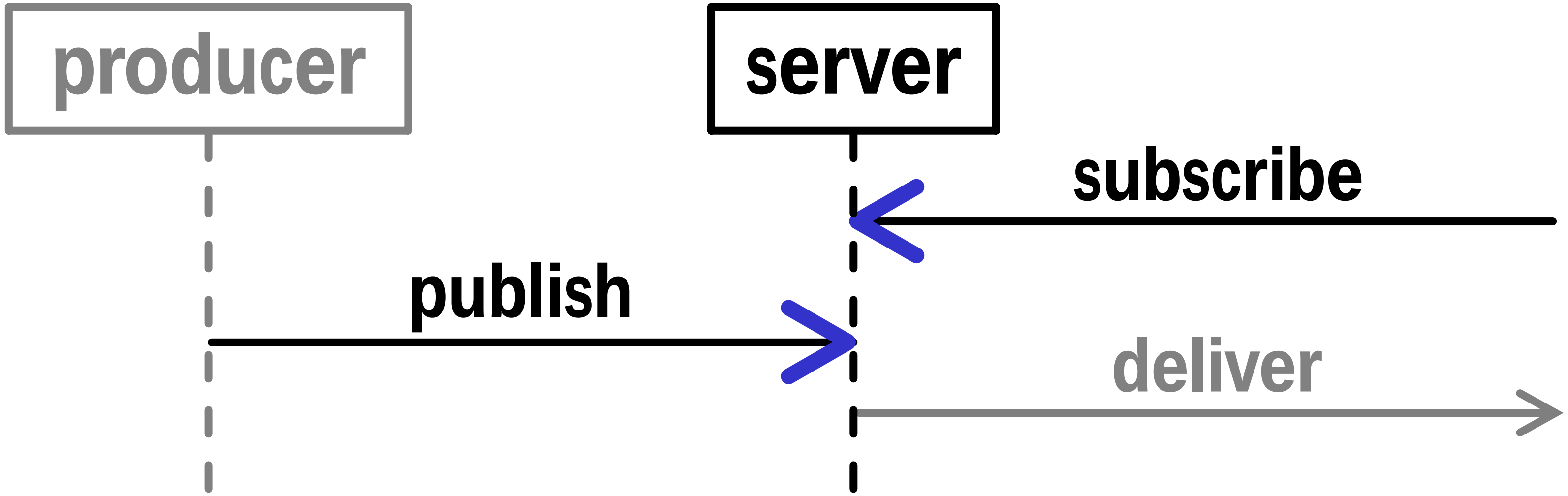


- = Message from sender required to match usage specs.

Overview

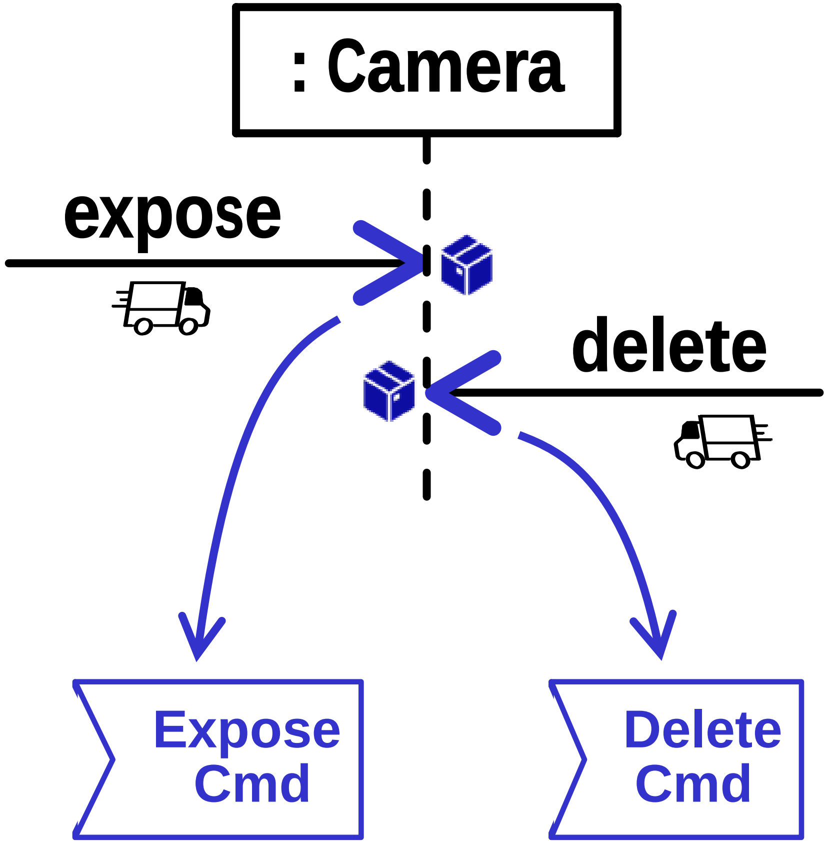
- Event handling, requirements
- Solutions, Kernel
  - Onto messages/flows
  - Transitions
  - Accepting “events”
- Solutions, SysML
  - Accept and send actions
  - **Sequence diagrams**
  - Flows
- Summary

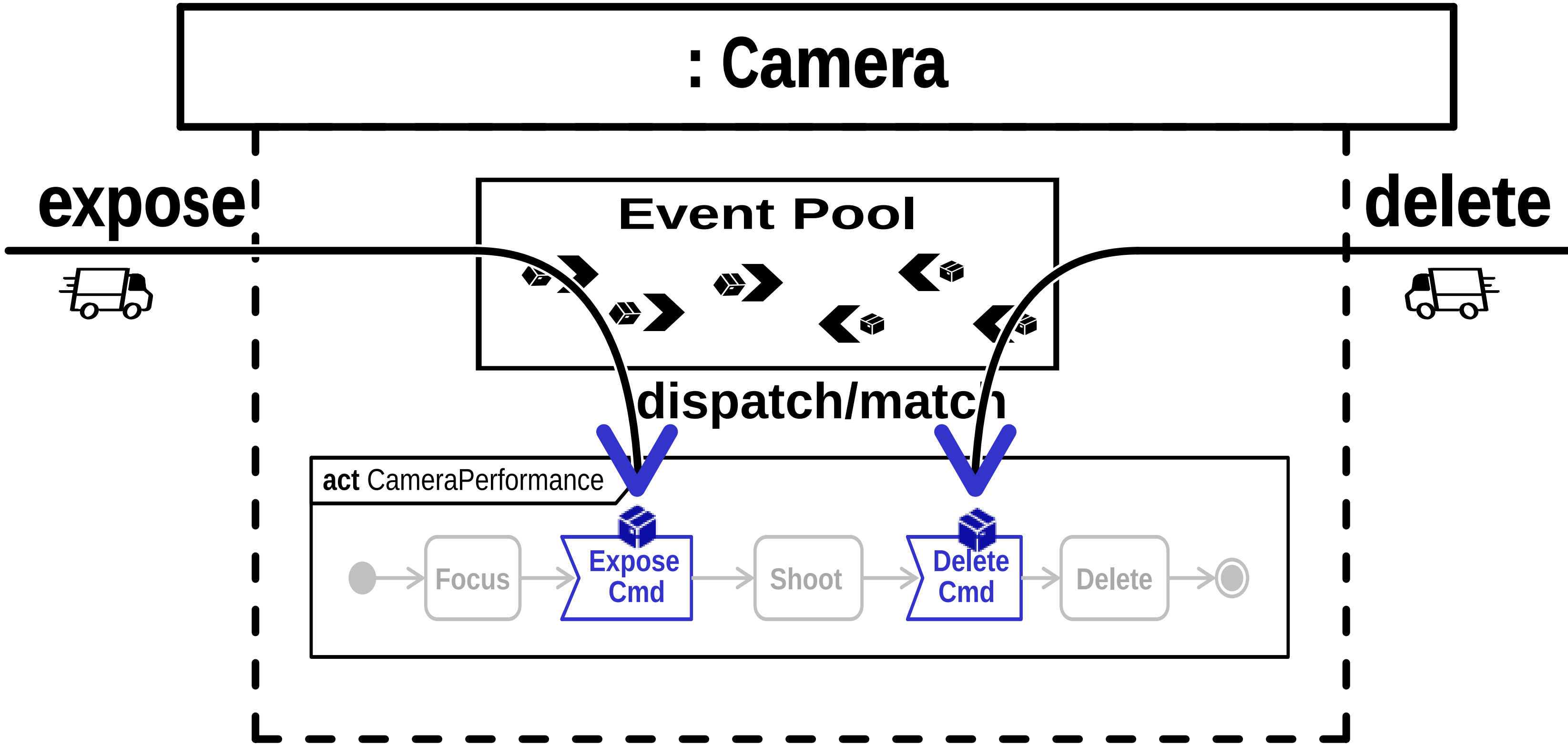
What do the arrowheads mean?



- Multiple possibilities …

Event Processing in Sequence Diagrams



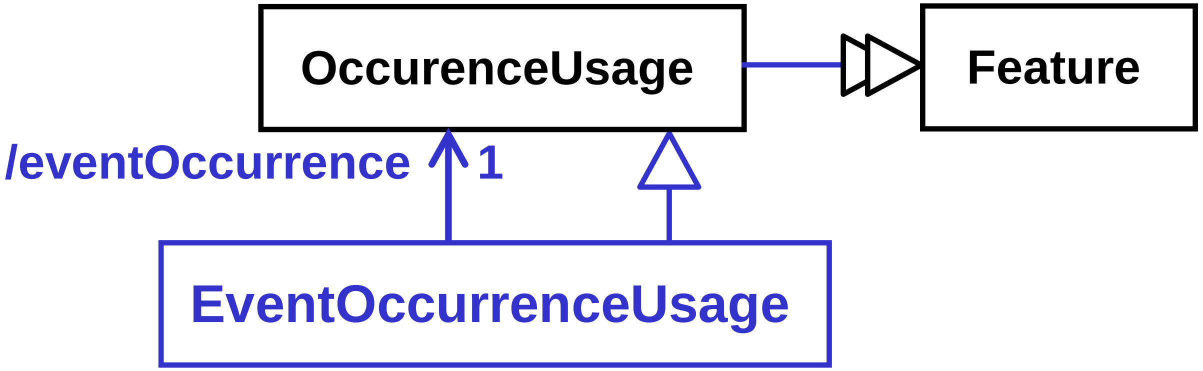


- **If messages end at accept …**
  - **… where does event processing fit in?**

SST Event Occurrences

- Anything happening during another occurrence
  - Presumably also inside the spatial region also.
- All other meaning is in how applications use them.
  - Eg, send/accept, arrive/leave, directed feature set/get.
- Only defined syntactically, not model library element.
- Intentionally don’t specify much.

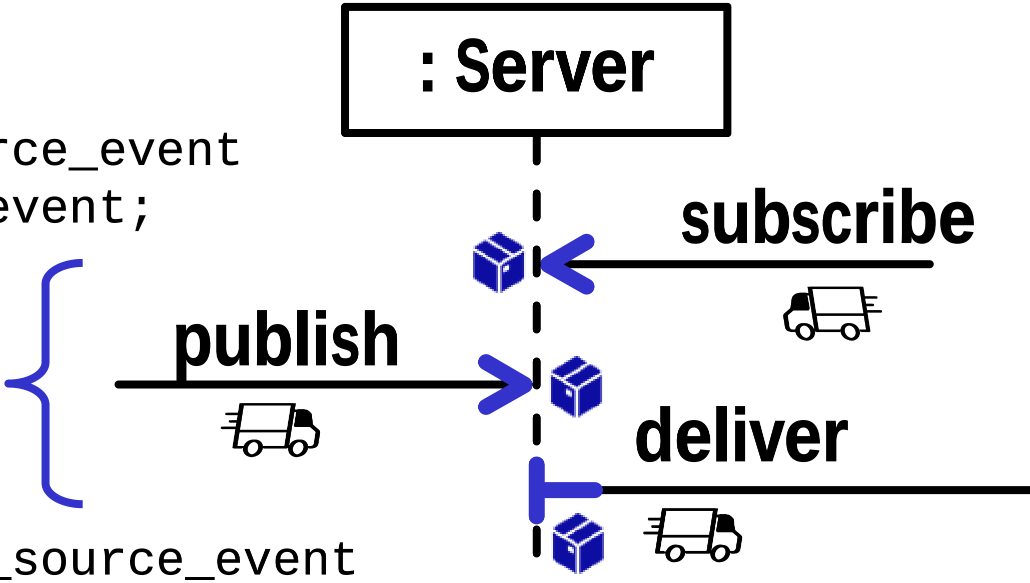
SysML
Metamodel
(M2)



Kernel
Metamodel
(M2)

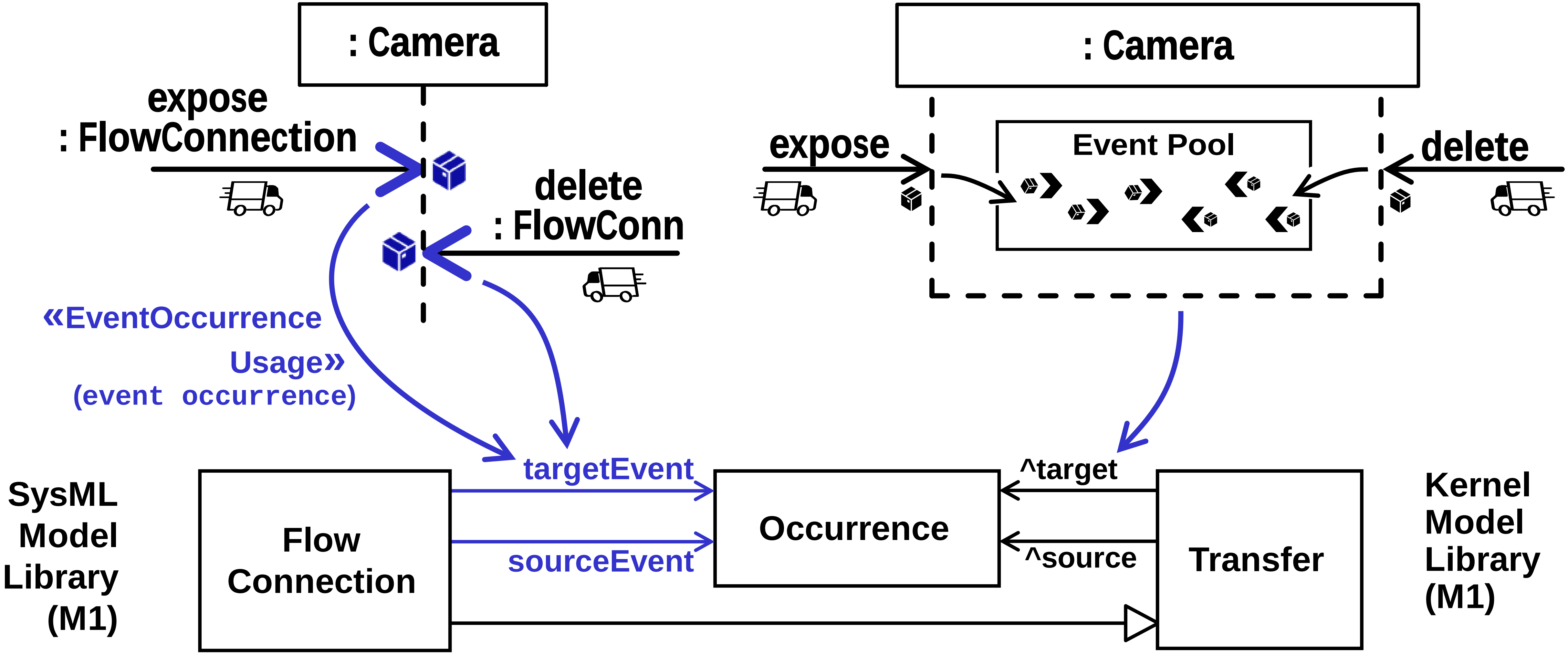
Sequence with Events

```
part def PubSubSequence {
  part producer[1] {
    event occurrence publish_source_event; }

  message publish_message from producer.publish_source_event
                          to server.publish_target_event;
  part server[1] {
    event occurrence subscribe_target_event;
    then event occurrence publish_target_event;
    then event occurrence deliver_source_event; }

  message subscribe_message from consumer.subscribe_source_event
                            to server.subscribe_target_event;
  message deliver_message from server.deliver_source_event
                          to consumer.deliver_target_event;
  part consumer {
    event occurrence subscribe_source_event;
    then event occurrence deliver_target_event; } }
```



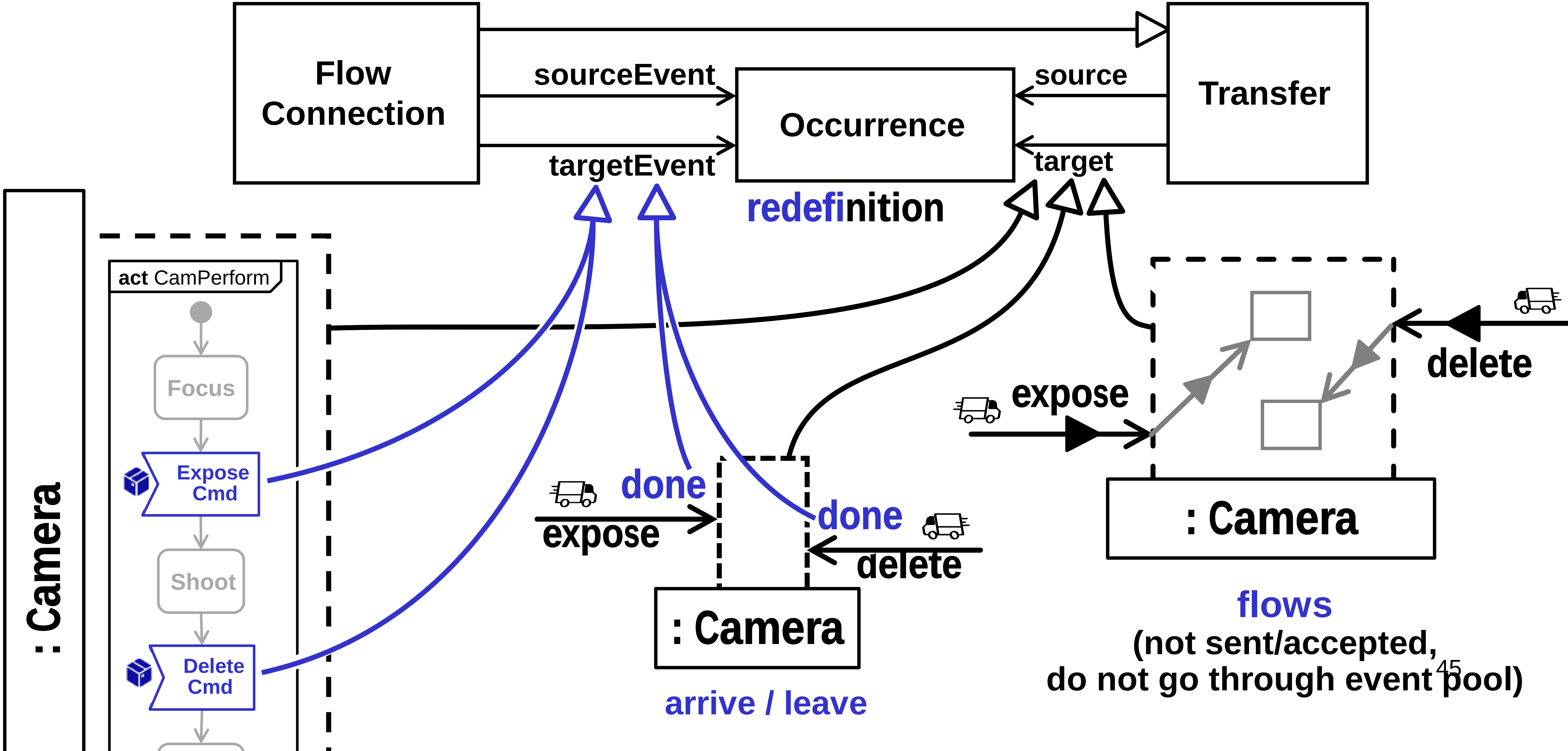
Not committing (yet) to what events are.

Flow Connection source/targetEvents



- Can happen before transfer starts and after it ends.

Multiple Interpretations of Events

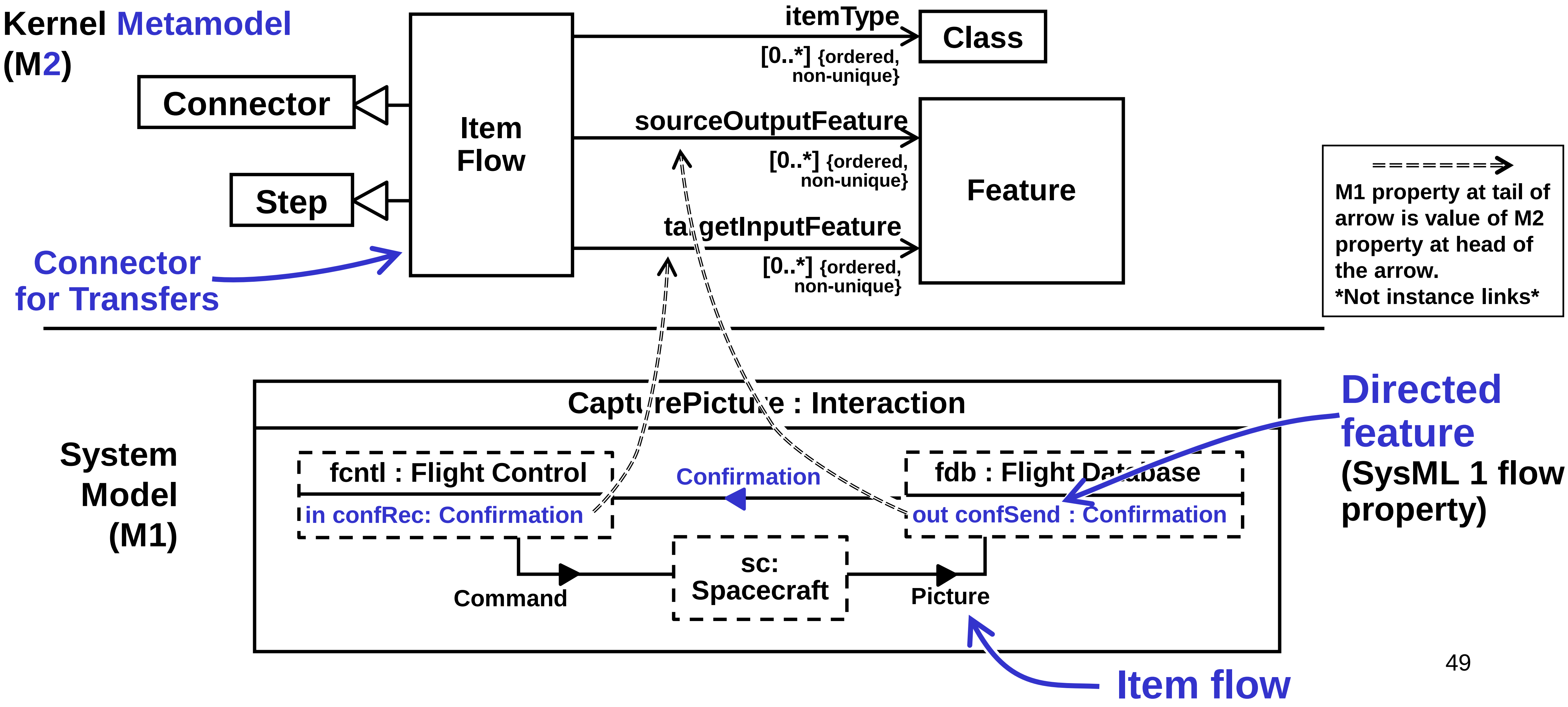


flows
(not sent/accepted,
do not go through event pool)

Overview

- Event handling, requirements
- Solutions, Kernel
  - Onto messages/flows
  - Transitions
  - Accepting “events”
- Solutions, SysML
  - Accept and send actions
  - Sequence diagrams
  - **Flows**
- Summary

Kernel Item Flows and Directed Features



Kernel Metamodel (M2)

Connector

Step

Item Flow

itemType [0..*] {ordered, non-unique}

Class

sourceOutputFeature [0..*] {ordered, non-unique}

targetInputFeature [0..*] {ordered, non-unique}

Feature

Connector for Transfers

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

System Model (M1)

CapturePicture : Interaction

fcntl : Flight Control

in confRec: Confirmation

Confirmation

fdb : Flight Database

out confSend : Confirmation

sc: Spacecraft

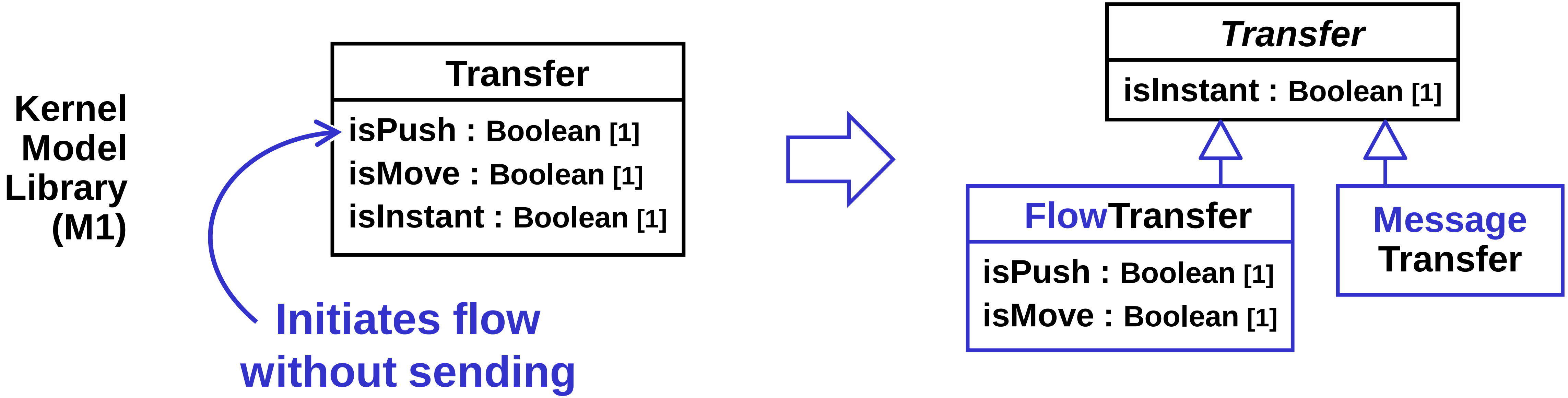
Command

Picture

Directed feature (SysML 1 flow property)

Item flow

Transfers for Messaging and Flows



Kernel
Model
Library
(M1)

Transfer

isPush : Boolean [1]
isMove : Boolean [1]
isInstant : Boolean [1]

Initiates flow
without sending

Transfer

isInstant : Boolean [1]

FlowTransfer

isPush : Boolean [1]
isMove : Boolean [1]

Message
Transfer

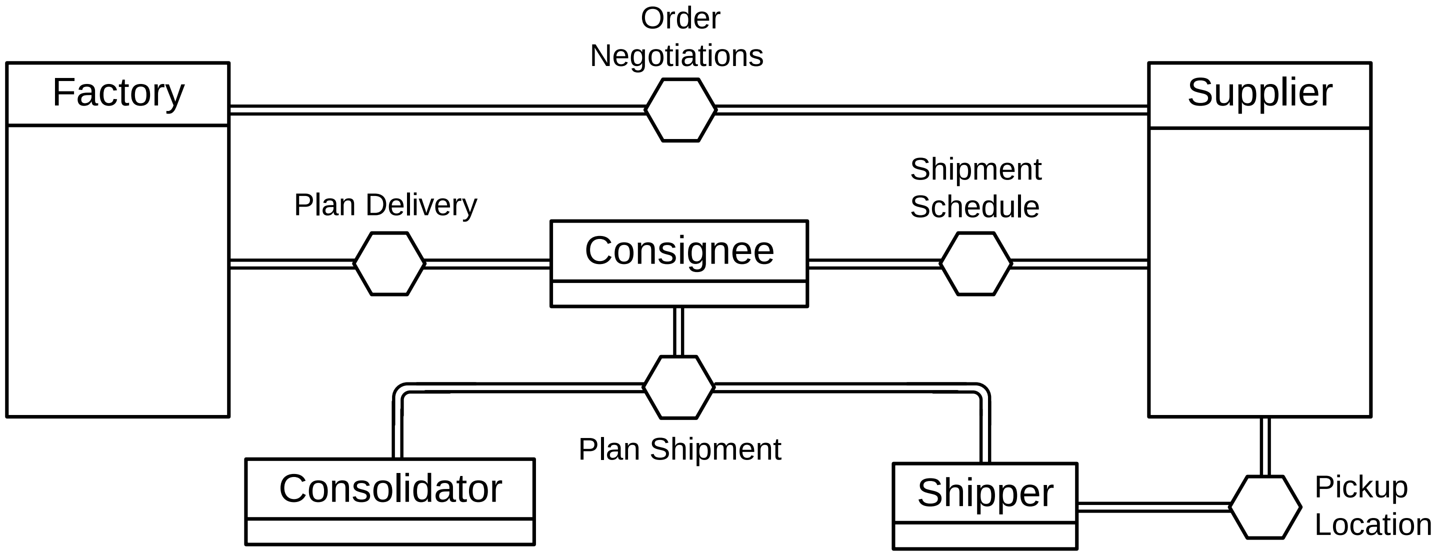
- **Separate out “flow” characteristics.**
  - **Messages don’t have them.**
- **TBD:**
  - **Coordinate terms (message keyword, FlowConnection)**
  - **Incoming transfers shouldn’t include flows?**
  - **Only a SysML distinction, not Kernel?**

TBD

- Completion “events”
- Deferral proposal, but not in slides
- Triggers on NonStateTransitions?
- Time/change event priority.

Forget about Event Processing?

- Even experts forget about it
  - Maybe it should be forgotten in general.
- Weak compared to BPMN “conversations”.
  - Messages identify the conversation they’re in.



New Capabilities for Interaction Modeling in BPMN 2.0, Bock & White, 2012. http://conradbock.org/bock-bpmn2-interactions-bookmark-web.pdf

Overview

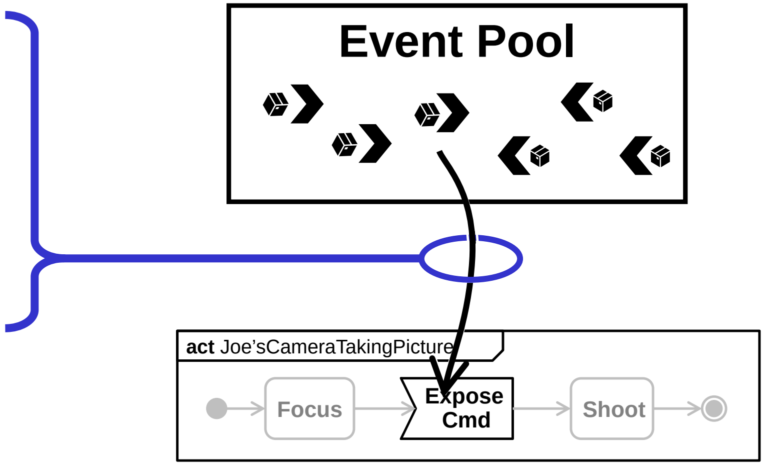
- Event handling, requirements
- Solutions, Kernel
  - Onto messages/flows
  - Transitions
  - Accepting “events”
- Solutions, SysML
  - Accept and send actions
  - Sequence diagrams
  - Flows
- **Summary**

Event Handling, Summary

- Objects managing reaction to incoming messages
  - Often within agreed interactions with other objects.
- Specified procedurally in current OMG standards.

RTC

1) Select event in pool
2) Match against usage specs
3) Evaluate guards
4) Act (maybe)



- SST proposal available:
  - Modeling conditions on valid traces.
  - Integrate with SST actions, sequence diagrams, and flows.

Current Proposal for Event Handling

- **Increased flexibility in**
  - **RTC (scope)**
  - **Dispatch (scope)**
  - **Pool location / prioritization**
- **Integrates event handling with**
  - **Accept actions**
  - **Sequence diagrams**
- **TBD**
  - **More triggers**
  - **Time/change event priority**