COLLEGE of CHARLESTON

Systems *Engineering: Design and Development*

ENGR 387

Agenda

- Introduction
- Purpose and Use of Sequence and Sequence Diagrams
- Sequence Diagram Elements
  - Lifelines
  - Event Ocurrences
- Fragment Types
- Cosntraints
- Purpose and Use of Actions
- Purpose and Use of Token Flow
- Purpose and Use of Object Flows
- Purpose and Use of Control Flows
- When Does an Action Start?
- When Does an Activity End?
- Review Questions

Introduction

- Objectives:
  - Understand the purpose and use of the Sequence Diagrams
  - Understand the purpose and use of all types of Nodes and Edges
  - Understand Token Flow
  - Understand when an Action starts

- Additional information can be obtained by reviewing:
  - SysML Distilled (Delligatti)
    - Chapter 7 Sequence Diagrams

Purpose and Use of Sequence and Sequence Diagrams

- Interaction: specifying a behavior when you want the focus to be on how the parts of a block interact with one another via operation calls and asynchronous signals to produce an emergent behavior
- Sequence diagram:
  - Focuses on interactions (communication between objects) with the behavior
  - Shows sequential order of INFORMATION FLOW
  - A sequence diagram describes the events for a simple interaction across all objects involved

- Activity diagram frame:
  - diagramKind abbreviation: *sd*
  - modelElementType: *lifeline*
  - Model Element Name: namespace of model element represented by the diagram

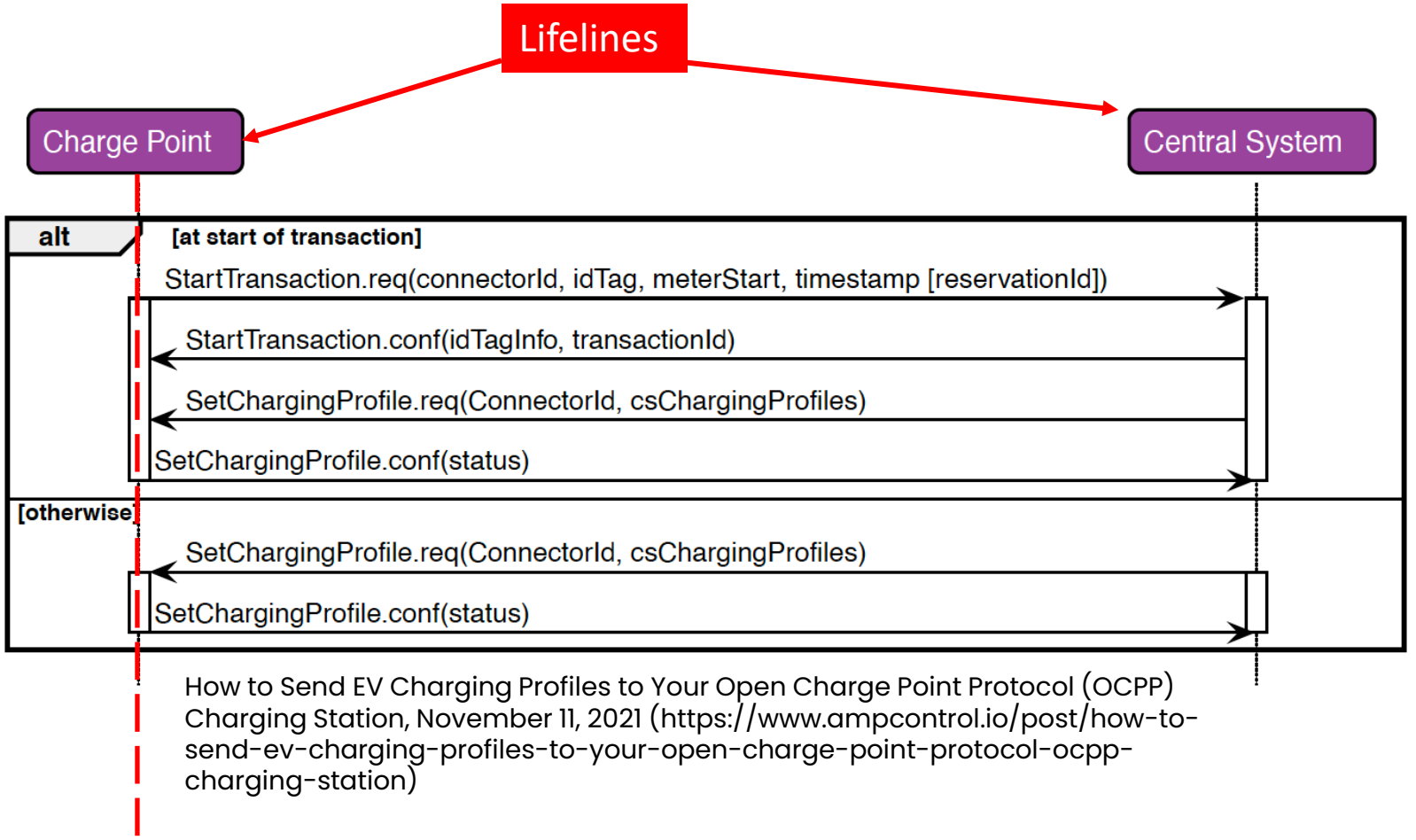
Sequence Diagrams Elements

- 7 kinds of elements found on a sequence diagram
  - Lifelines
  - Event Occurrences
  - Messages
  - Execution Specifications
  - Traces
  - Constraints
  - Combined Fragments

Sequence Diagram Elements – Lifelines

- Lifelines
  - Represents a SINGLE part property in a block, elsewhere in the model…
    - Syntax: <part property name> [<selector expression>] : type
    - Represents a participant in the interaction, therefore it represents a structure of some kind
    - Time proceeds DOWN the lifeline (Tail of the lifeline represents the relative passage of time)
    - A lifeline represents a single instance of its type (never multiple instances)

Sequence Diagram Elements – Lifelines (cont.)



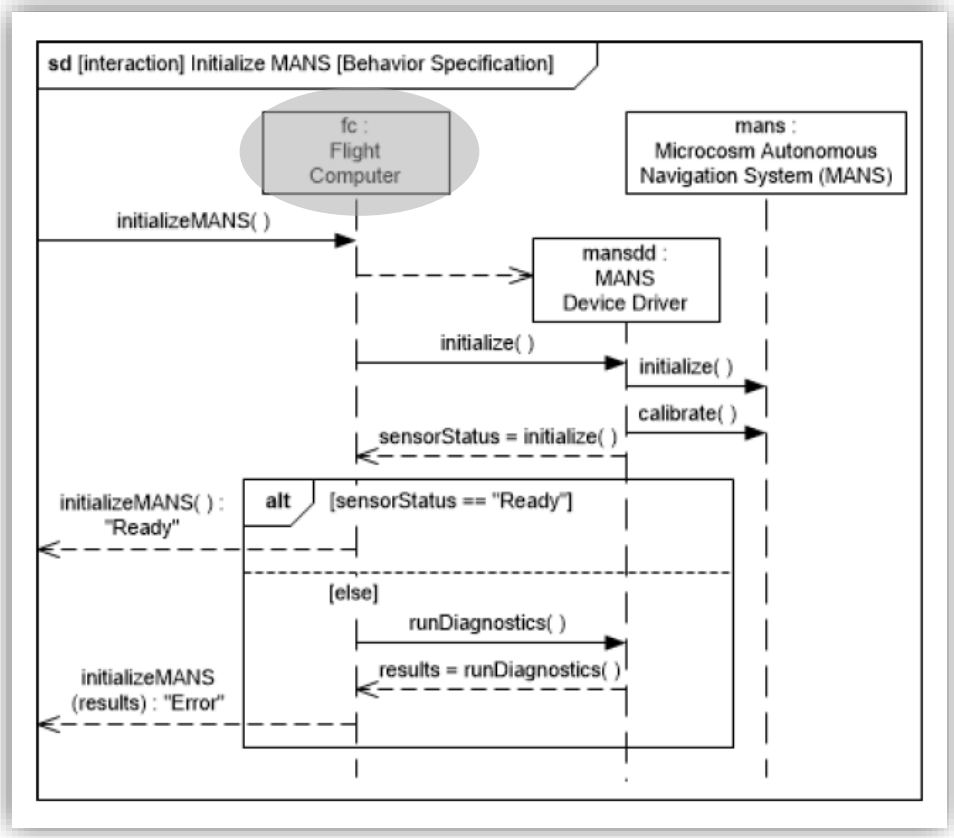
How to Send EV Charging Profiles to Your Open Charge Point Protocol (OCPP) Charging Station, November 11, 2021 (https://www.ampcontrol.io/post/how-to-send-ev-charging-profiles-to-your-open-charge-point-protocol-ocpp-charging-station)

Represent the role that takes part to the interaction

COLLEGE of CHARLESTON
1770

Sequence Diagram Elements – Lifelines (cont.)

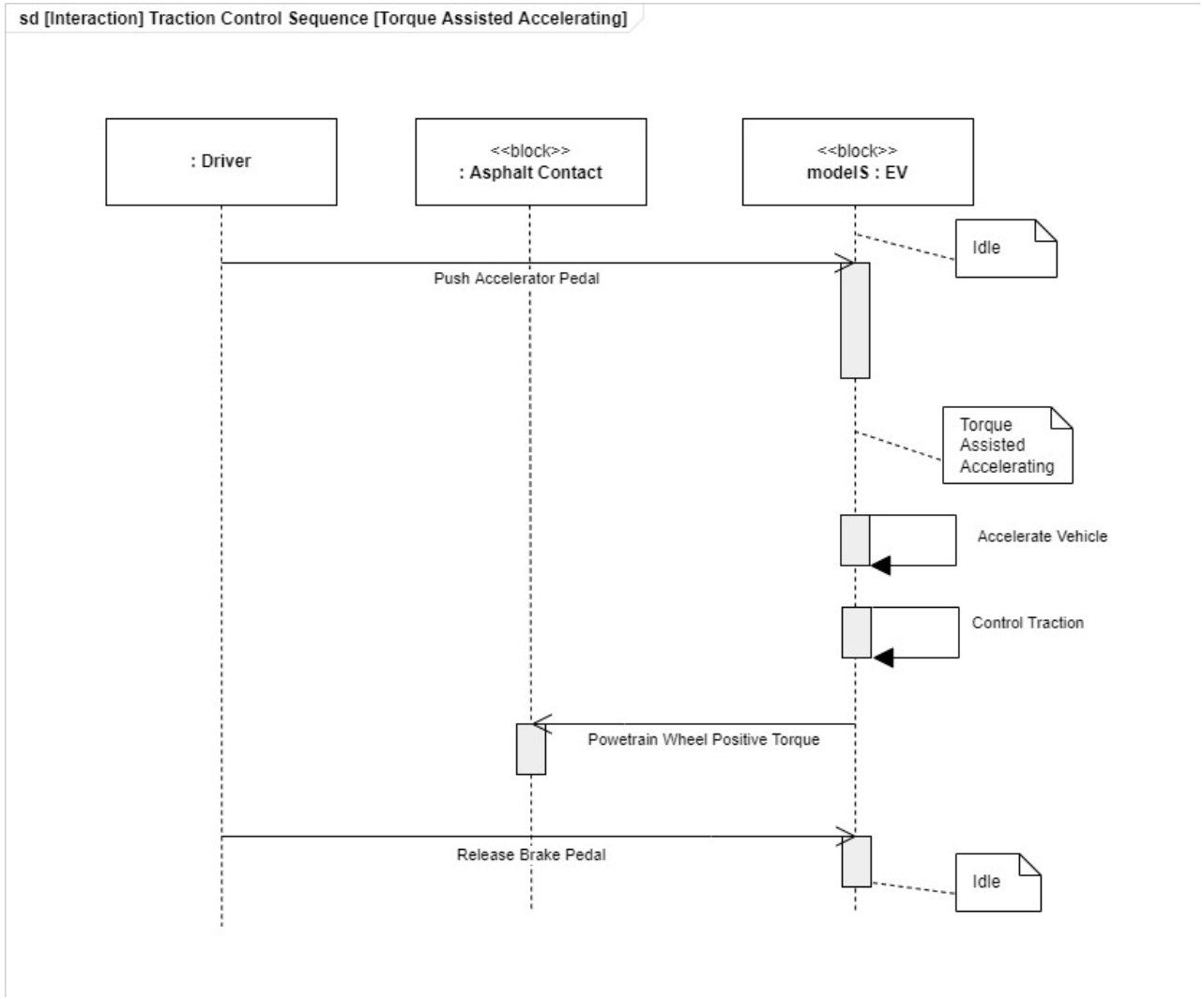
- Lifeline
  - Part Property = fc
  - Type = Flight Computer
  - The block that owns the life-line, also owns all the "receptions" along that life-line



[3] Images used, courtesy of Delligatti & Associates,LLC

Sequence Diagram Elements – Question 1: Lifelines

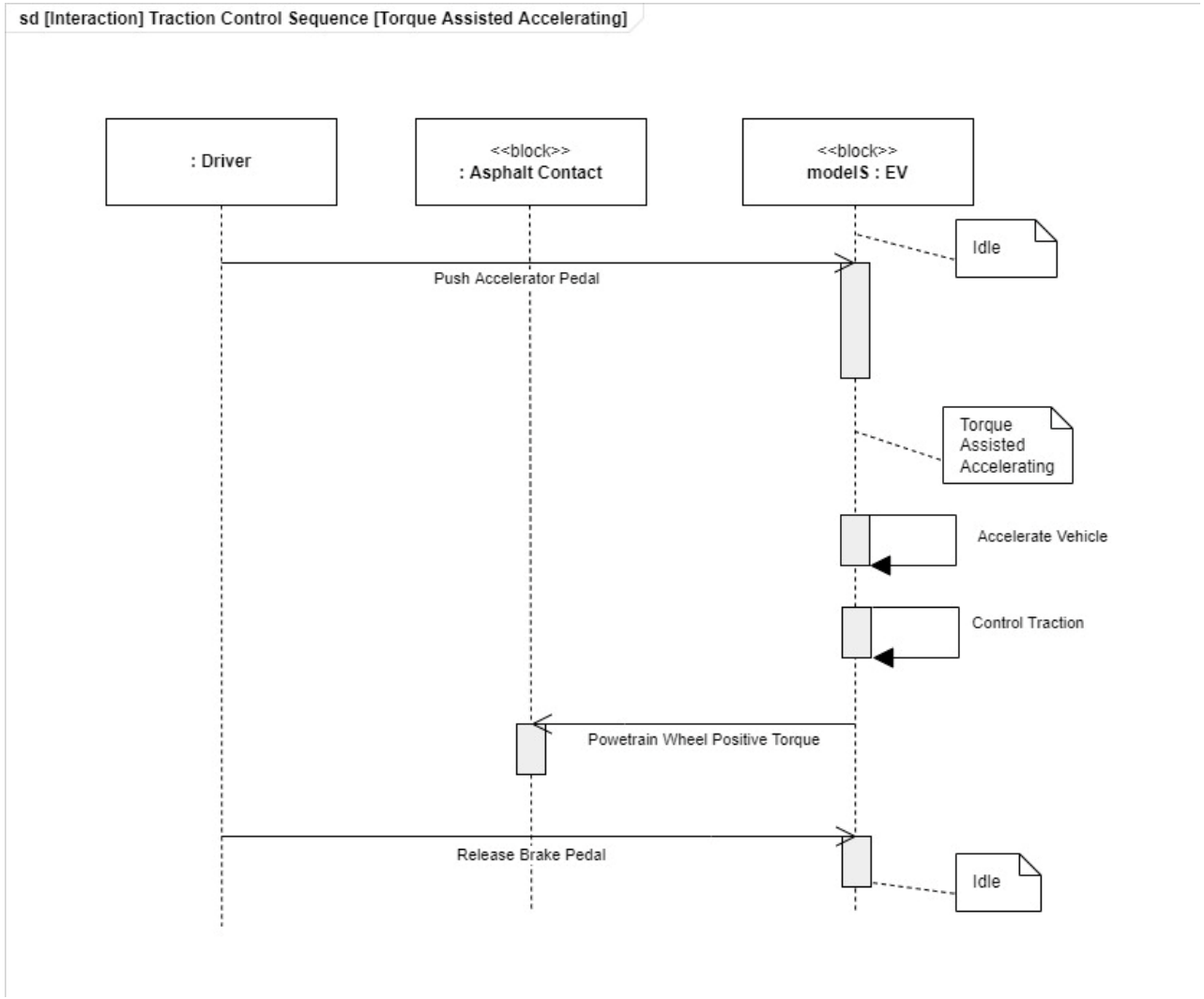
- Which of the following is the correct description, based on the diagram?
  - *a)* *Driver* is not a valid lifeline, because it is an actor, not a part property
  - *b)* *Asphalt Contact* is not a valid lifeline, because it is a block, not a part property
  - *c)* *modelS : EV* is not a valid lifeline because it is an instance of a block, and not a part property
  - d) All three of these lifelines are valid.



COLLEGE of CHARLESTON
1770

Sequence Diagram Elements – Question 1: Lifelines

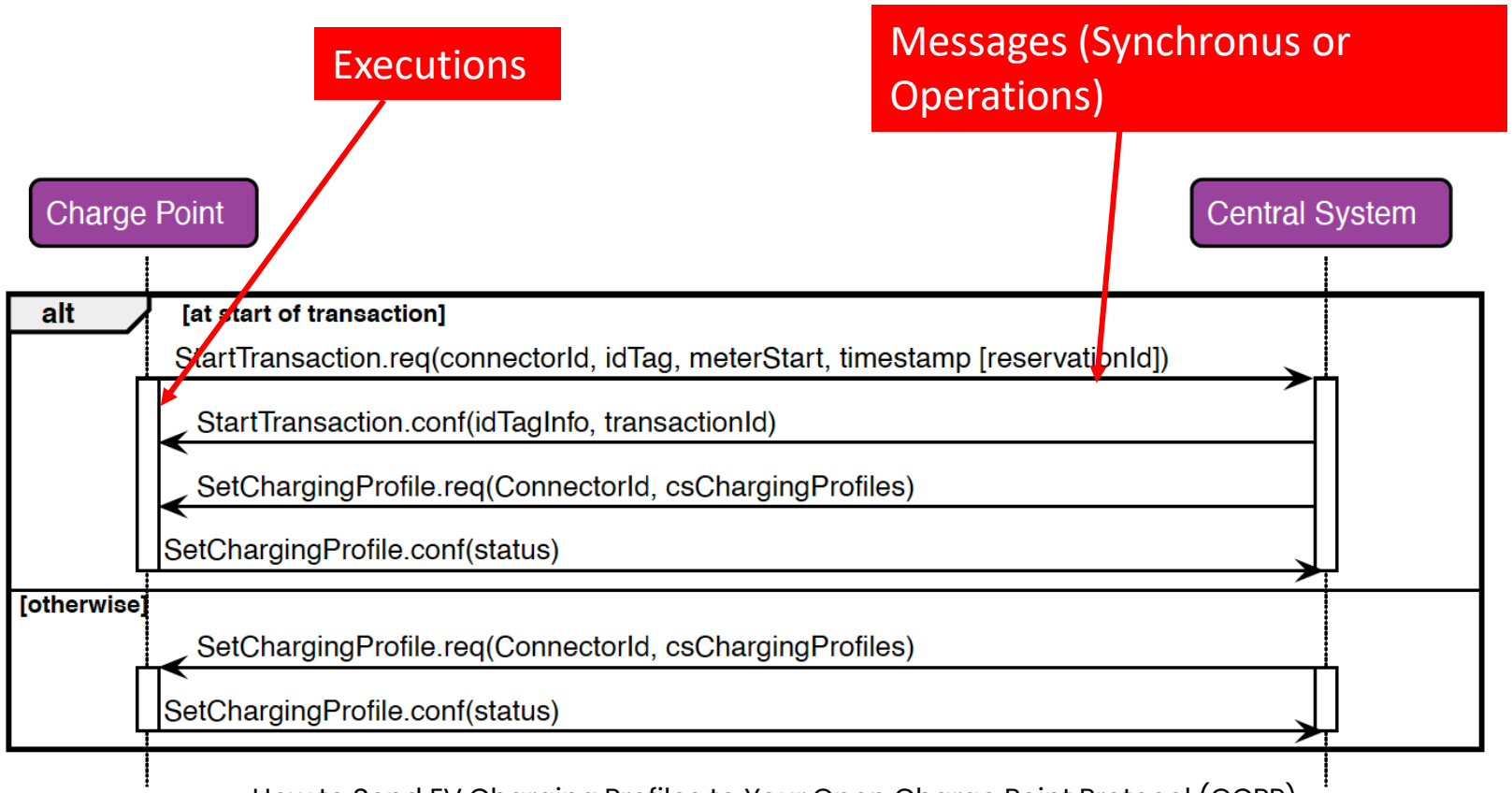
- Which of the following is the correct description, based on the diagram?
  - *a)* *Driver* is not a valid lifeline, because it is an actor, not a part property
  - *b)* *Asphalt Contact* is not a valid lifeline, because it is a block, not a part property
  - *c)* *modelS : EV* is not a valid lifeline because it is an instance of a block, and not a part property
  - **d) All three of these lifelines are valid.**



COLLEGE of CHARLESTON
1770

Sequence Diagram Elements (cont)

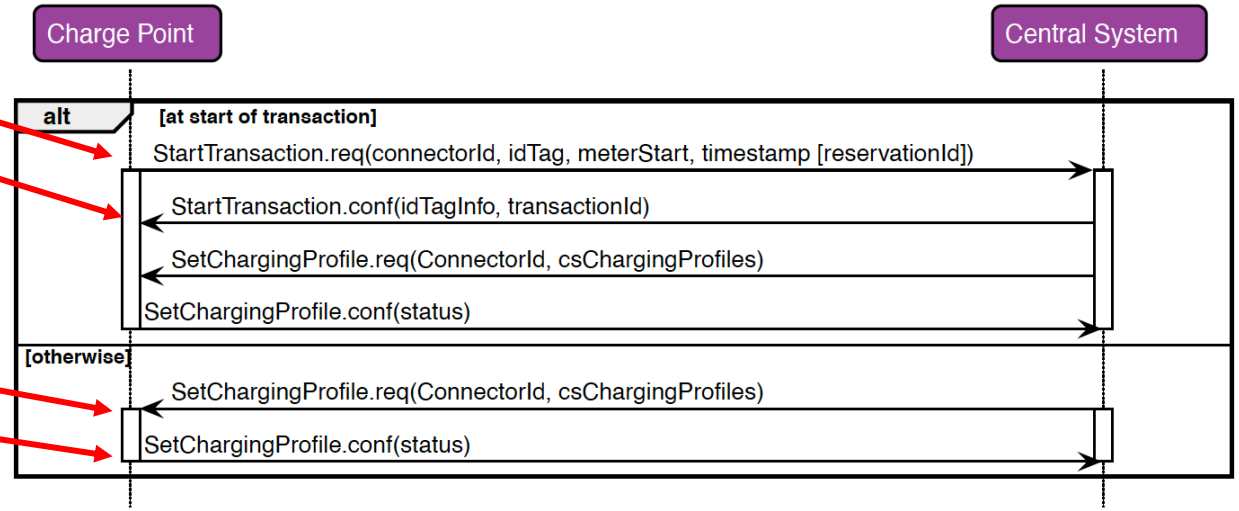
Identify sections of the diagram with special semantics (as defined by interaction operators)



How to Send EV Charging Profiles to Your Open Charge Point Protocol (OCPP) Charging Station, November 11, 2021 (https://www.ampcontrol.io/post/how-to-send-ev-charging-profiles-to-your-open-charge-point-protocol-ocpp-charging-station)

Sequence Diagram Elements – Event Occurrences

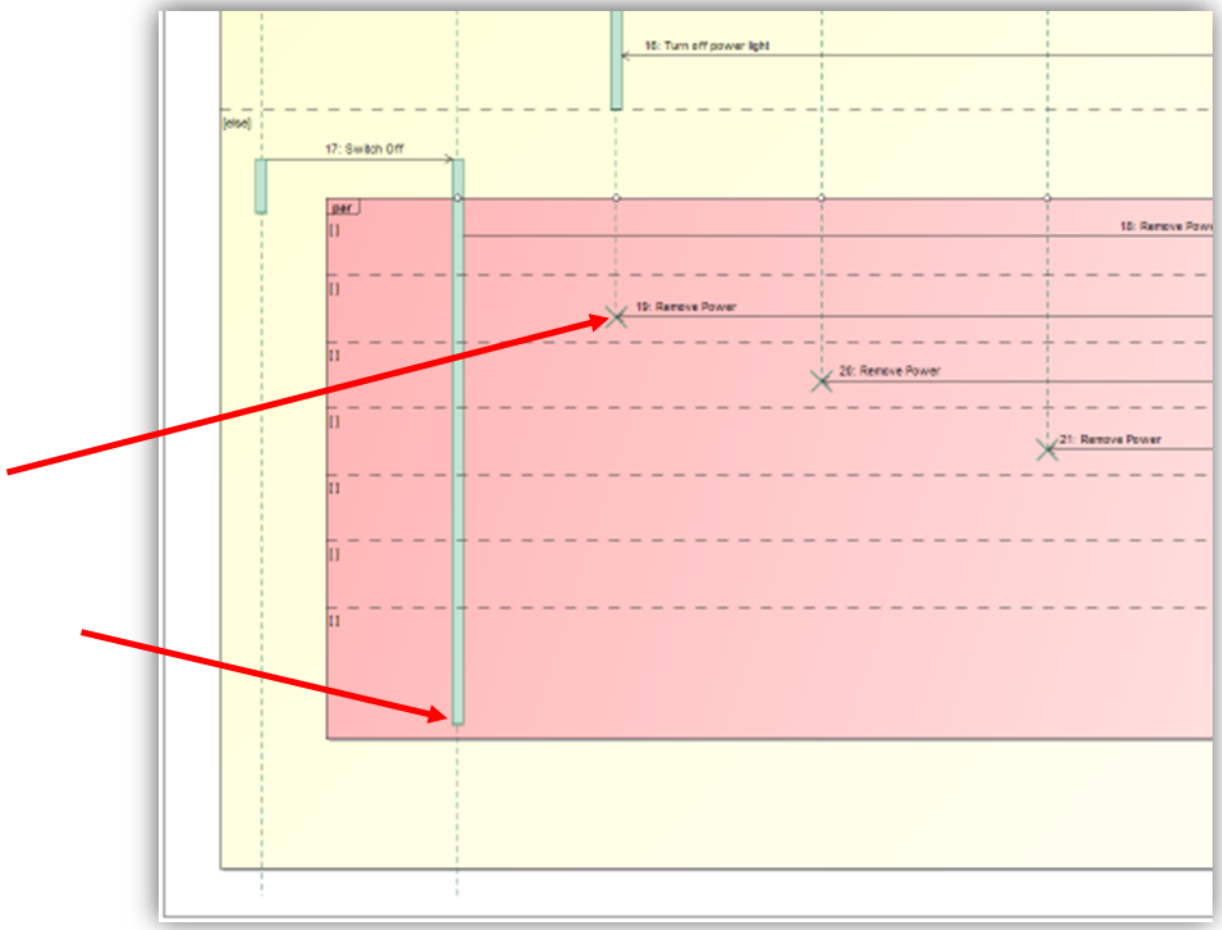
- Event Occurances
  - 6 types
    - Message Send
    - Message Receive
    - Lifeline Create
    - Lifeline Destruct
    - Behavior Start
    - Behavior Terminate



How to Send EV Charging Profiles to Your Open Charge Point Protocol (OCPP) Charging Station, November 11, 2021 (https://www.ampcontrol.io/post/how-to-send-ev-charging-profiles-to-your-open-charge-point-protocol-ocpp-charging-station)

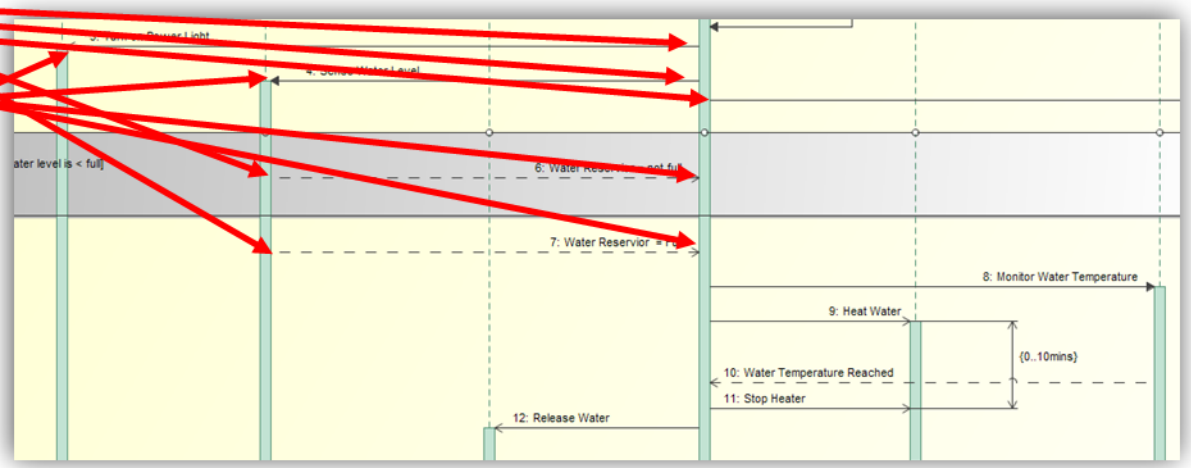
Sequence Diagram Elements – Event Occurrences (cont.)

- Event Occurances
  - 6 types
    - Message Send
    - Message Receive
    - Lifeline Create
    - Lifeline Destruct
    - Behavior Start
    - Behavior Terminate



Sequence Diagram Elements – Event Occurrences (cont.)

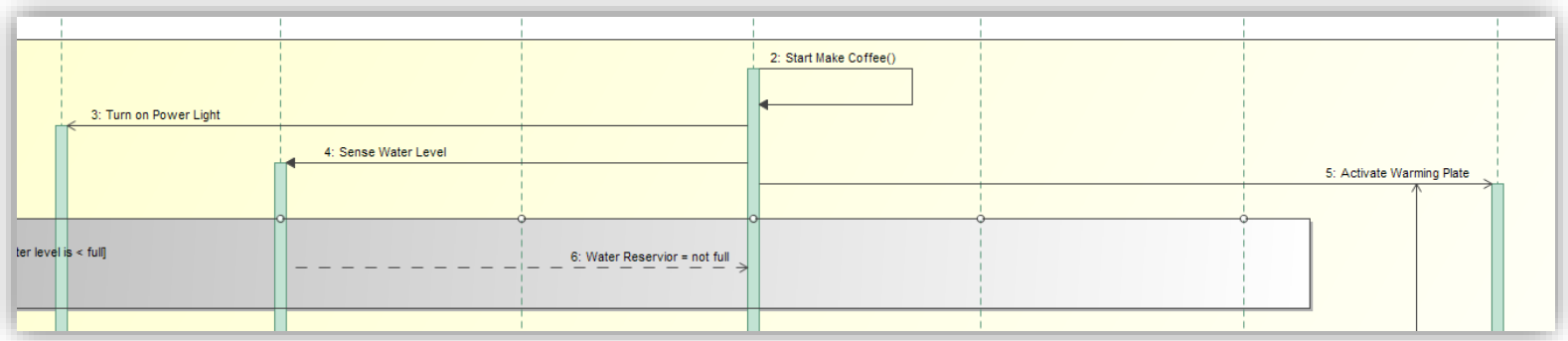
- "Message Occurance" includes both…
  - Message Send
  - Message Receive



- 'Message' in SysML, simply conveys a "communication" between elements, not that data is being passed
  - *Message* ***DOES NOT = DATA***)

Sequence Diagram Elements – Question 1: Event Occurrences

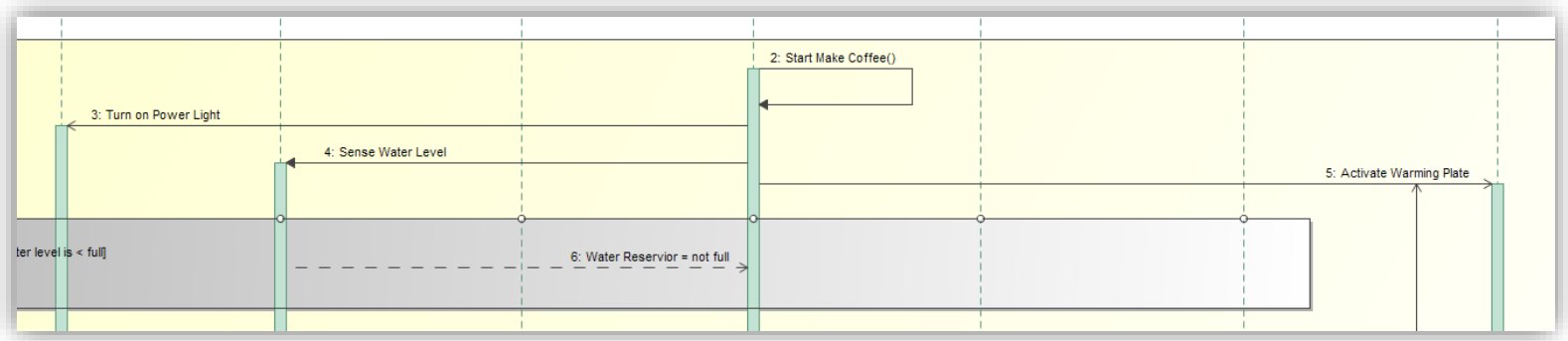
- How many message occurrences are shown on this sequence diagram?



a) 5
b) 6
c) 10
d) 11

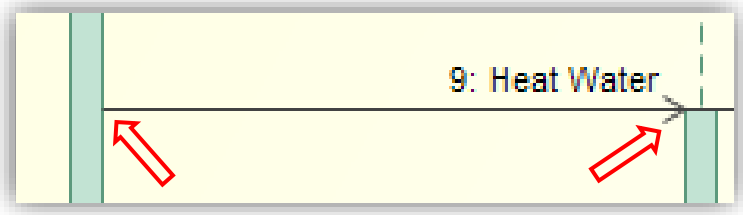
Sequence Diagram Elements – Question 1: Event Occurrences

- How many message occurrences are shown on this sequence diagram?

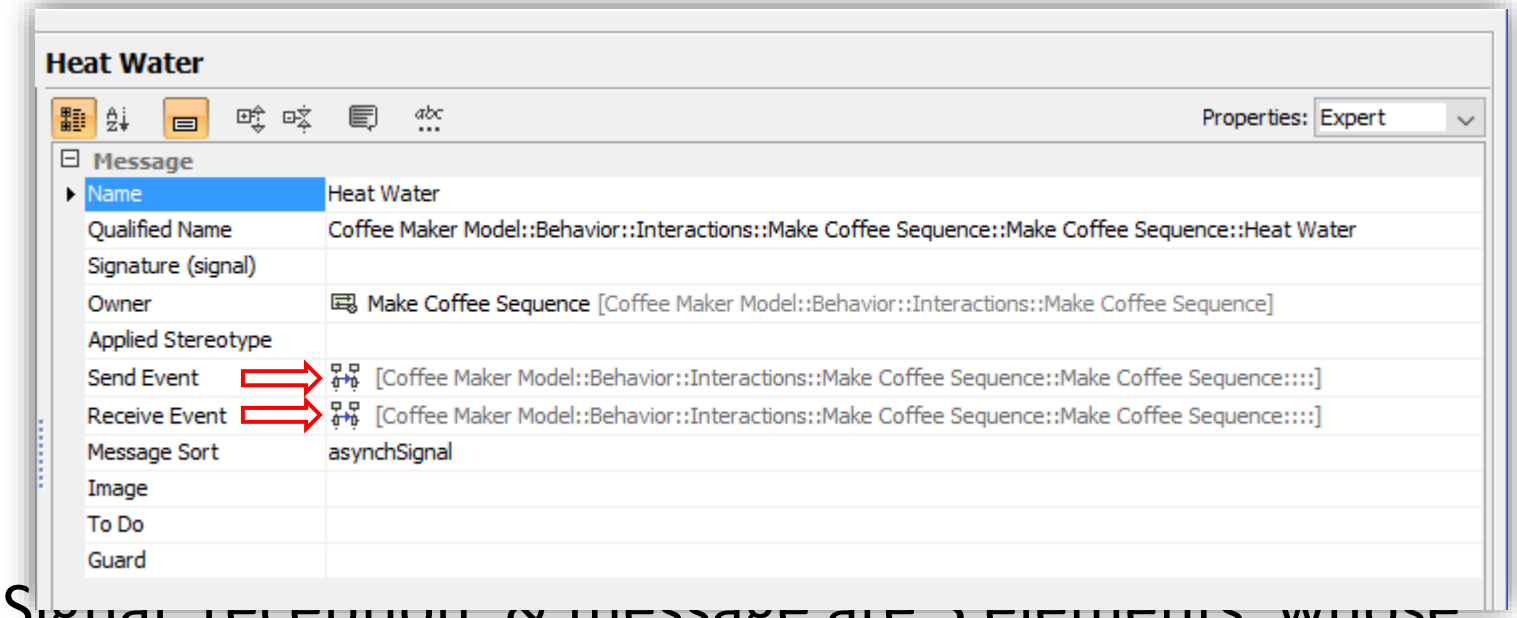


a) 5
b) 6
c) 10
d) 11

Asynchronous Messages

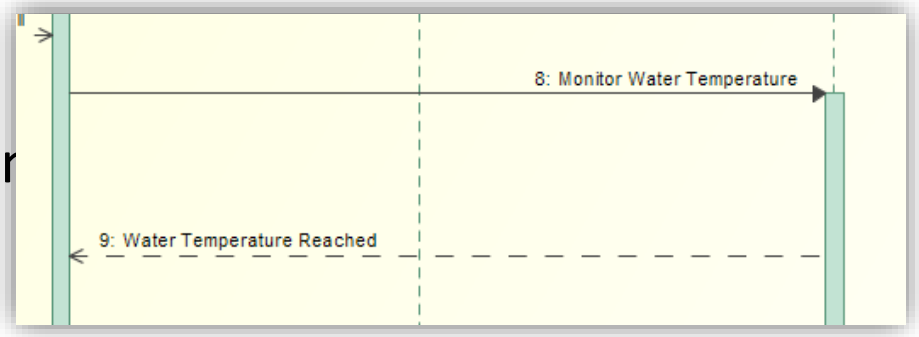


- Solid line with empty arrow on receiving end
- Message name and reception name must match, but are two different elements in the model



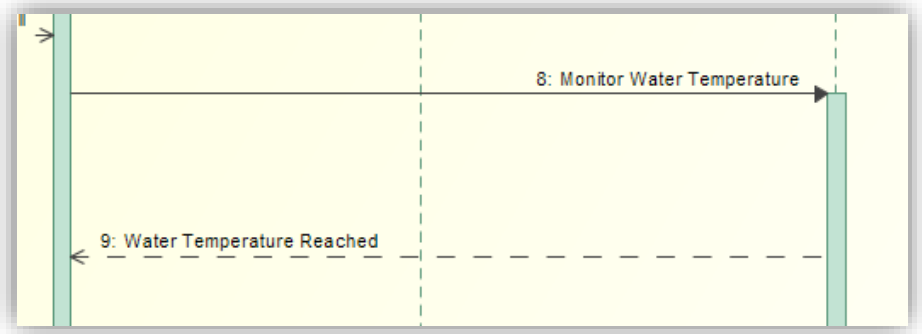
- Signal, reception, & message are 3 elements, whose names must match…

Synchronous Message



- Solid line with filled in arrow or receiving end
- REMEMBER that a synchronous message always has a reply, even if it isn't shown on the diagram
- Represents a call to an operation on the lifeline at the receiving end (receiving lifeline (block) owns operation)
- The client element that sends the message, (which calls the operation), waits for the behavior to complete and the completed reply message to be received

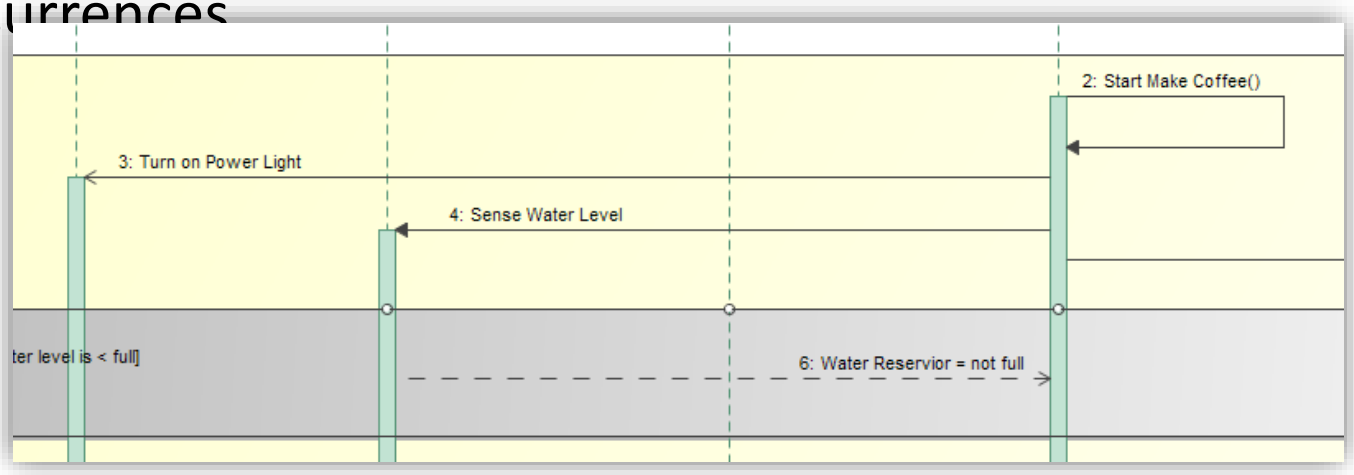
Synchronous - Reply Messages



- Reply Messages are a dashed line with an open arrowhead
- Reply Messages (only synchronous) are sent back to the caller that sent the sync message when the operation is complete
- Optional to display the name of the reply message, name has to match the sync message name (can contain a return value)
- Reply message can be shown optionally, but is not required to be displayed for every sync message

Sequence Diagrams Question 3: Message Types

- Identify the correct statement for the following message occurrences



a) 2 = asynchronous message
3 = synchronous message
4 = asynchronous message
6 = synchronous message

b) 2 = synchronous message
3 = asynchronous message
4 = synchronous message
6 = asynchronous message

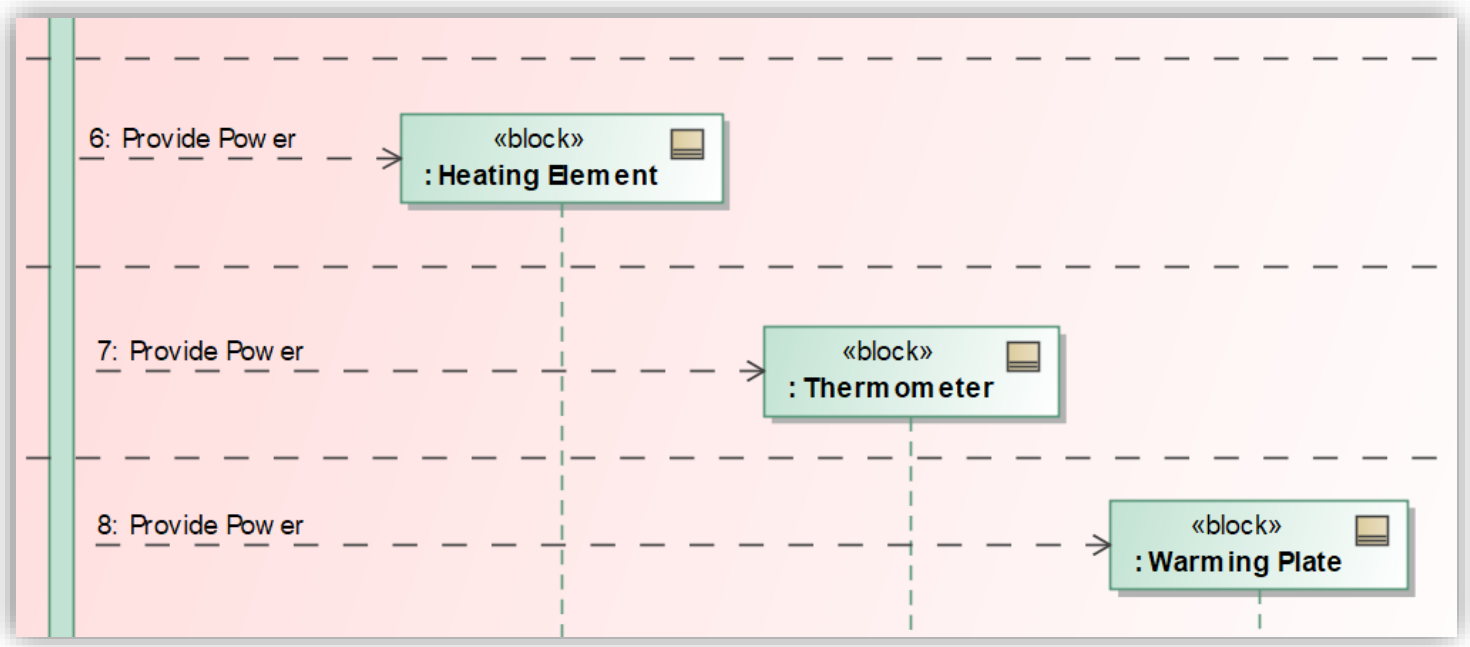
c) 2 = asynchronous message
3 = synchronous message
4 = asynchronous message
6 = reply message

d) 2 = synchronous message
3 = asynchronous message
4 = synchronous message
6 = reply message

Create Messages

- A dashed line with open arrowhead, which touches the head of a newly created lifeline

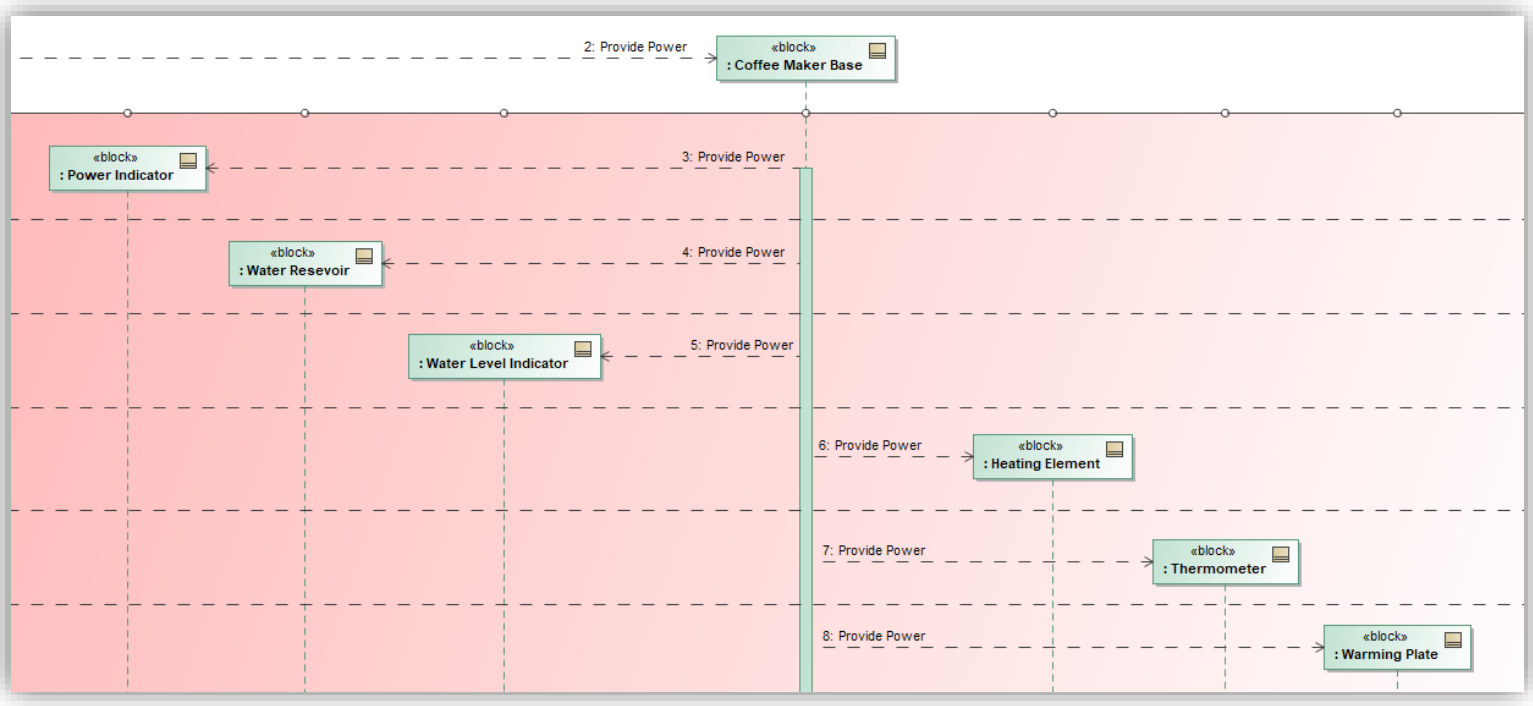
- Create message can have labels, but it does not affect the model operation (text only)



Sequence Diagrams Question 4: Lifeline Create Occurences

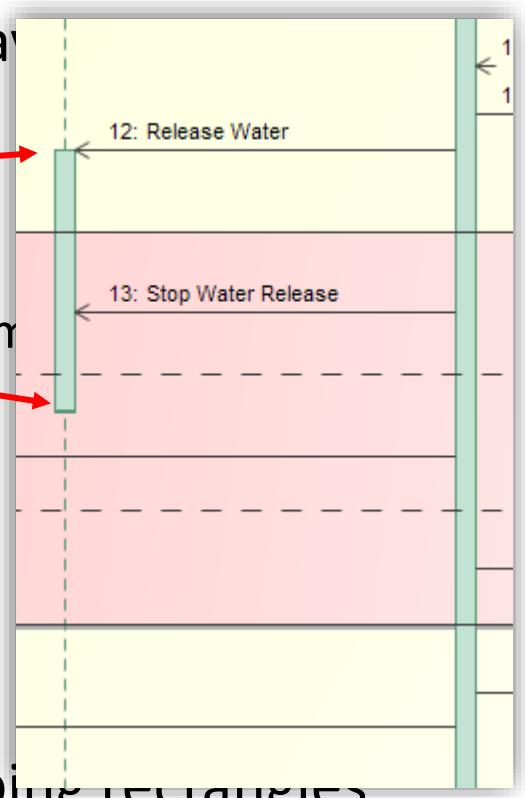
- How many lifeline create occurrences are specified in the following diagram?

a) 0
b) 3
c) 6
d) 7



Execution Specification

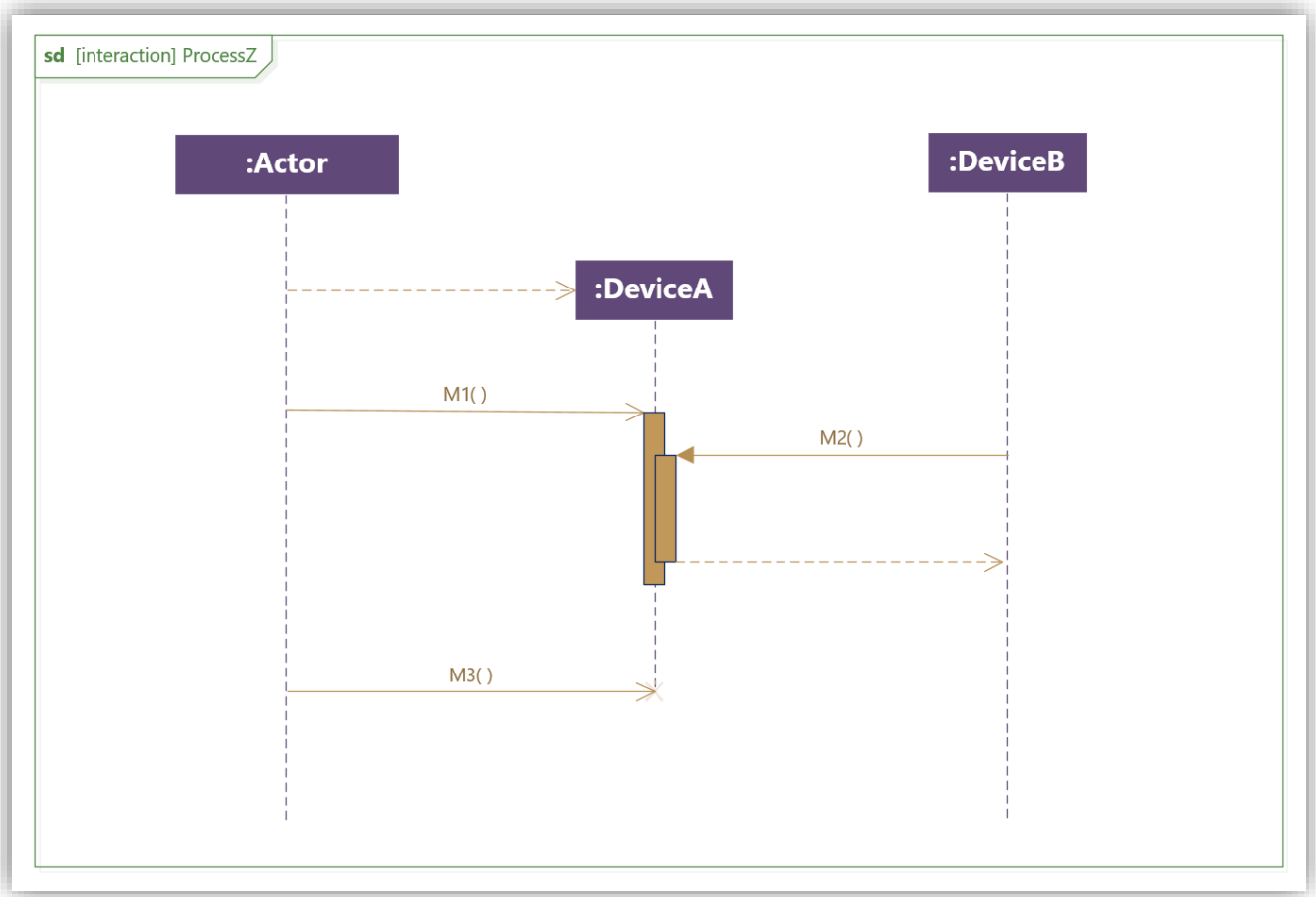
- Execution Specification is the "box" that wraps around the lifeline, and represents the execution of a beha
  - It has two (2) occurances…
    - Execution Start
    - Execution Finish
  - Showing it on a diagram is not required, but is assum to exist between receive and reply messages
- NOTE: There are 2 event occurrences at the point where an execution specification begins
  - Message Receive
  - Execution Specification Start
- Nested behaviors can be conveyed by overlapping rectangles on the same lifeline on a diagram



Sequence Diagrams Question 5: Event Occurrences

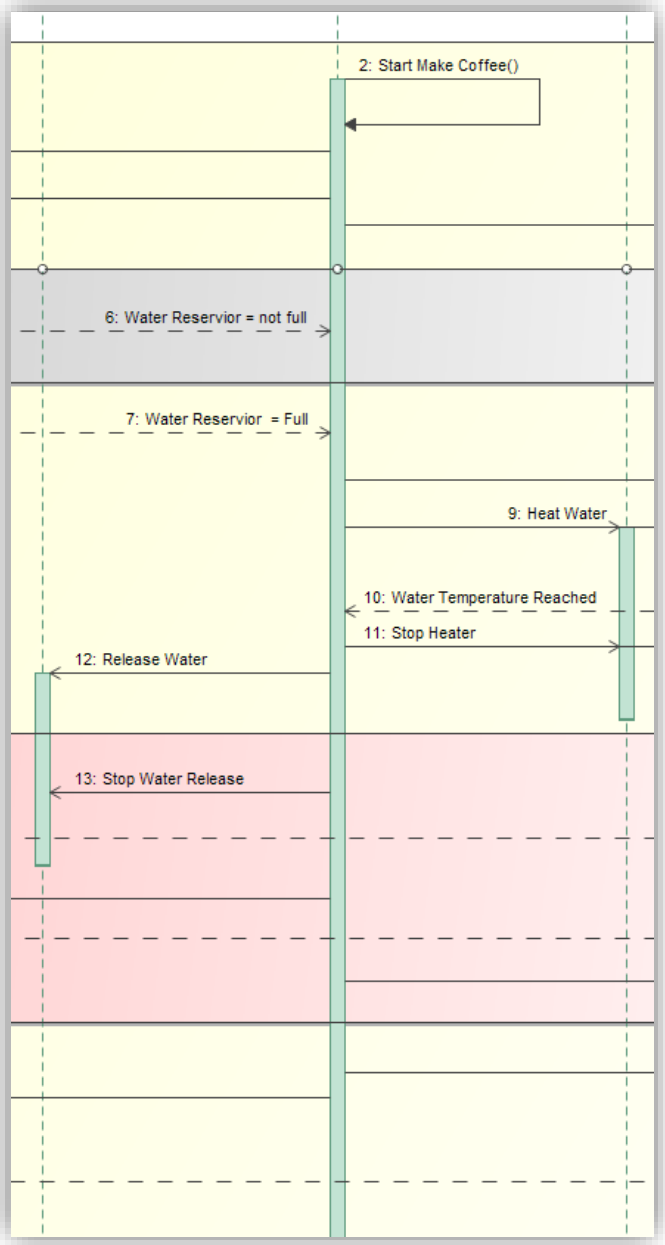
- How many event occurrences are specified in *ProcessZ Transaction?*

a) 3
b) 5
c) 7
d) 9
e) 10
f) 14
g) 16
h) 18



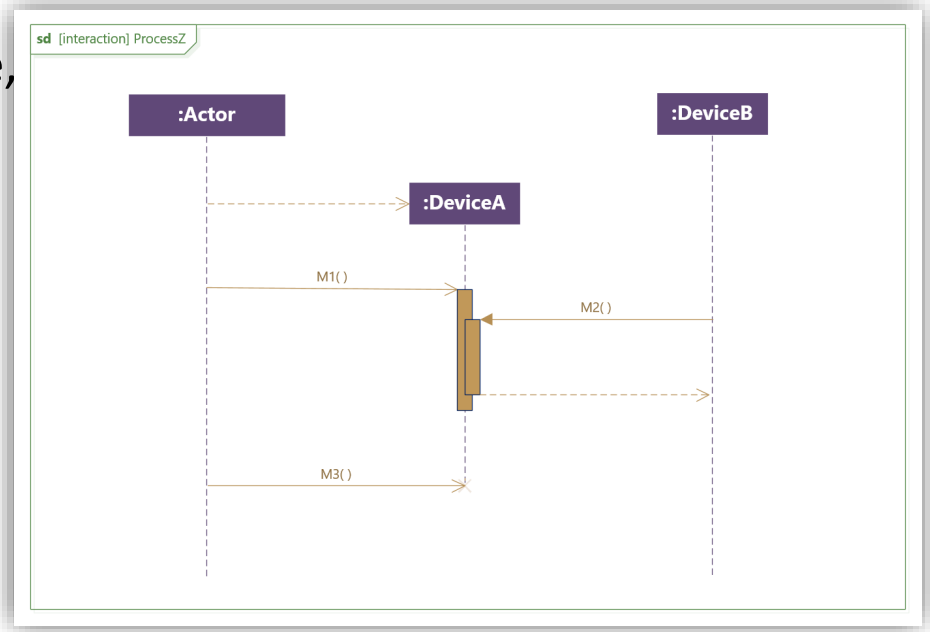
Traces

- 2 lifelines are not equivalent in "time“
  - Activities of one are not the same timing of the other)
- Reminder: Do not assume that separate timelines have order relationships, aside from message interactions and the associated rules…
  - Order down the lifeline has to be observed
  - Sending must occur before receipt



Sequence Diagrams Question 6: Trace

- Which of the following is an INVALID trace for *PRocessZ*?
  - Lifeline Create Message Send,
    Lifeline Create Message Receive,
    Lifeline Create…*then:*



a) M1.Send, M1.Receive, M2.Send, M2.Receive, M3.Send, M3.Receive, M2.Reply.Send, M2.Reply.Receive

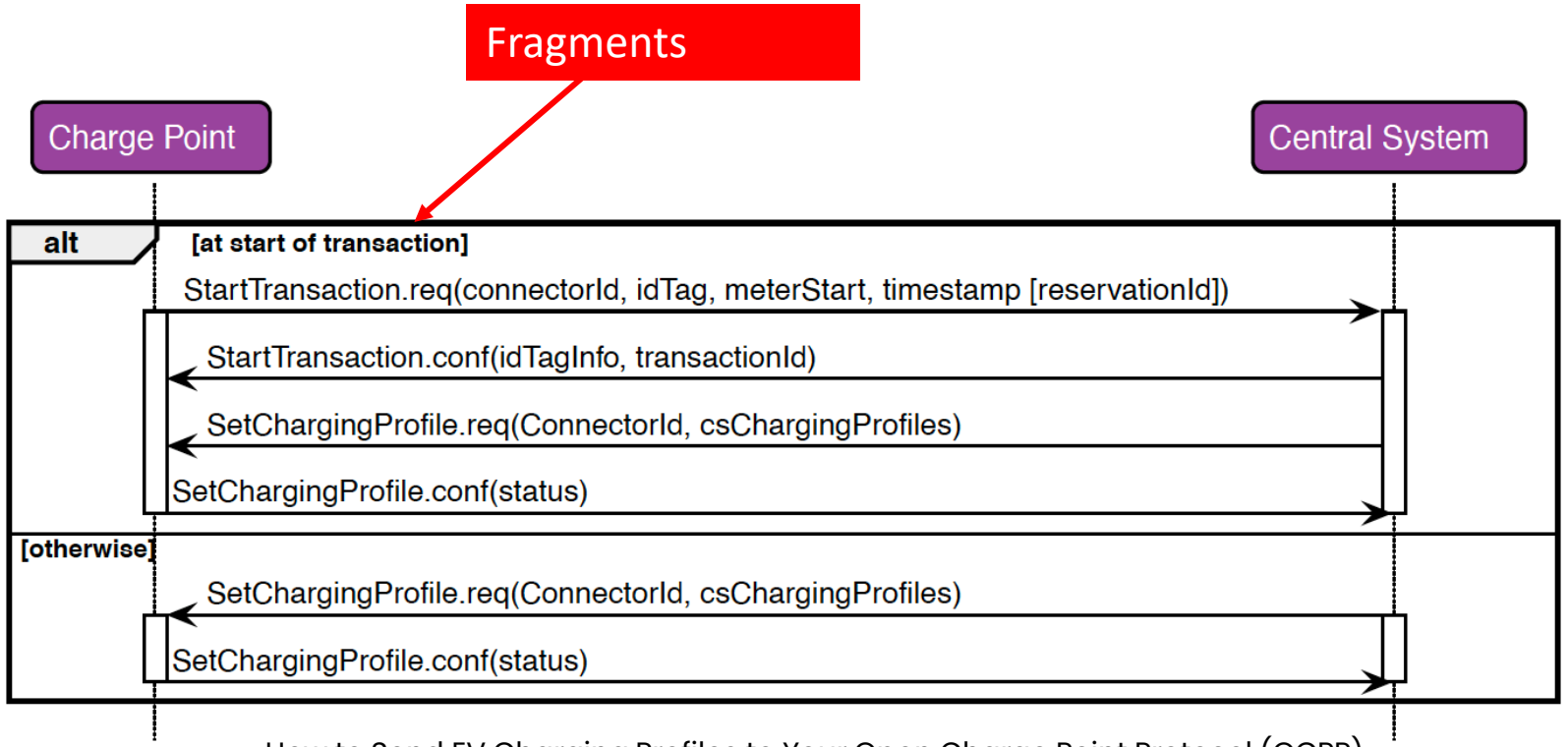
b) M1.Send, M1.Receive, M2.Send, M2.Recieve, M2.Reply.Send, M3.Send, M3.Receive, M2.Reply.Receive

c) M1.Send, M2.Send, M1.Receive, M2.Recieve, M2.Reply.Send, M2.Reply.Recieve, M3.Send, M3.Receive

d) M1.Send, M1.Receive, M2.Send, M2.Receive, M2.Reply.Send, M2.Reply.Recieve, M3.Send, M3.Receive

Sequence Diagram Elements (cont.)

Identify sections of the diagram with special semantics (as defined by interaction operators)



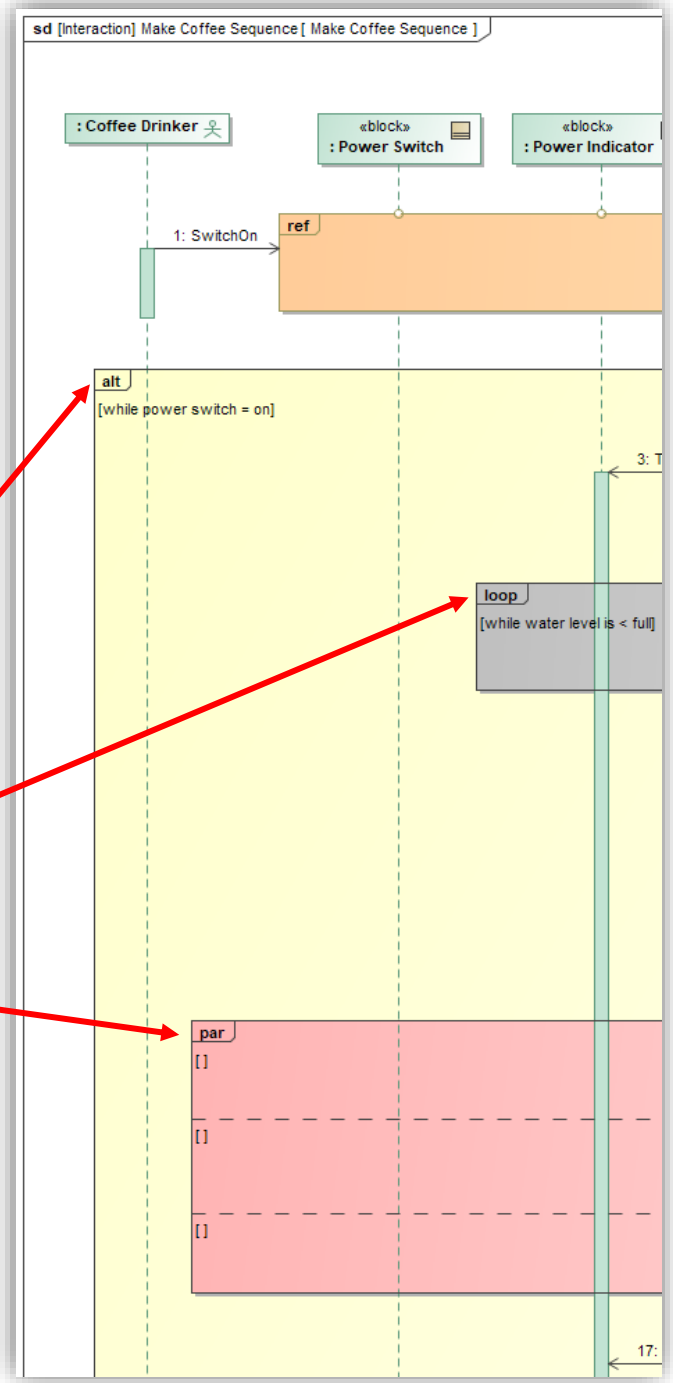
How to Send EV Charging Profiles to Your Open Charge Point Protocol (OCPP) Charging Station, November 11, 2021 (https://www.ampcontrol.io/post/how-to-send-ev-charging-profiles-to-your-open-charge-point-protocol-ocpp-charging-station)

Fragment Types

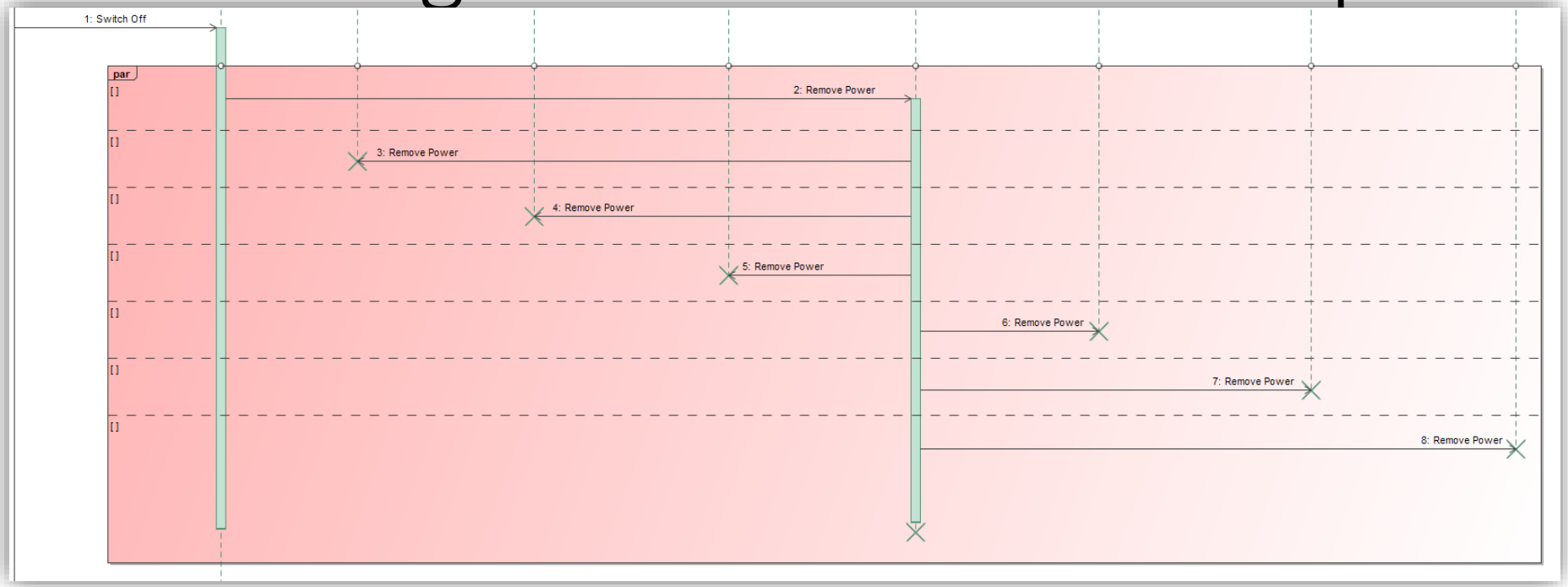
- **seq** (weak, the default) Each lifeline may see different orders for the exchange (subject to causality)
- **Strict.** The message exchange occurs in the order described
- **critical** The sequence diagram fragment is a critical region. It is treated as atomic – no interleaving with parallel regions.
- **neg** The sequence diagram fragment is forbidden. Either it is impossible to occur, or it is the intent of the requirements to prevent it from occurring.
- **assert** The sequence diagram fragment is the only one possible (or legal)
- **ref** name reference to a sequence diagram fragment defined elsewhere
- **opt** [condition] has 1 part that may be executed based on a condition/state value
- **alt** has 2 or more parts, but only one executes based on a condition/state and operand fragment labeled [else] is executed if no other condition is true
- **par** has 2 or more parts that execute concurrently (the order is undetermined)
- **loop** min..max [escape] Has a minimum # of executions, and optional maximum # of executions, and optional escape condition

Combined Fragments

- Combined Fragments are shown as rectangles with an interaction operator
- It is a mechanism that allows you to add control logic
  - (Group a subset of event occurances)
- Kinds of Combined Fragments
  - OPT, ALT, LOOP, PAR
    - (also 3 others not used on OCSMP levels 1&2)
- Inside the rectangle header is the Interaction Operator



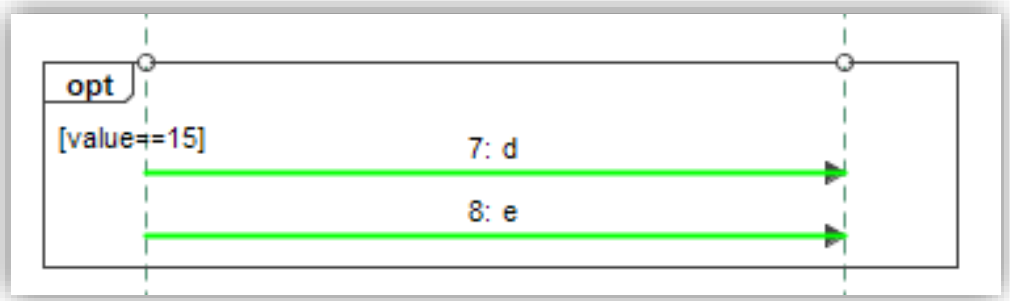
Combined Fragments – Interaction Operands



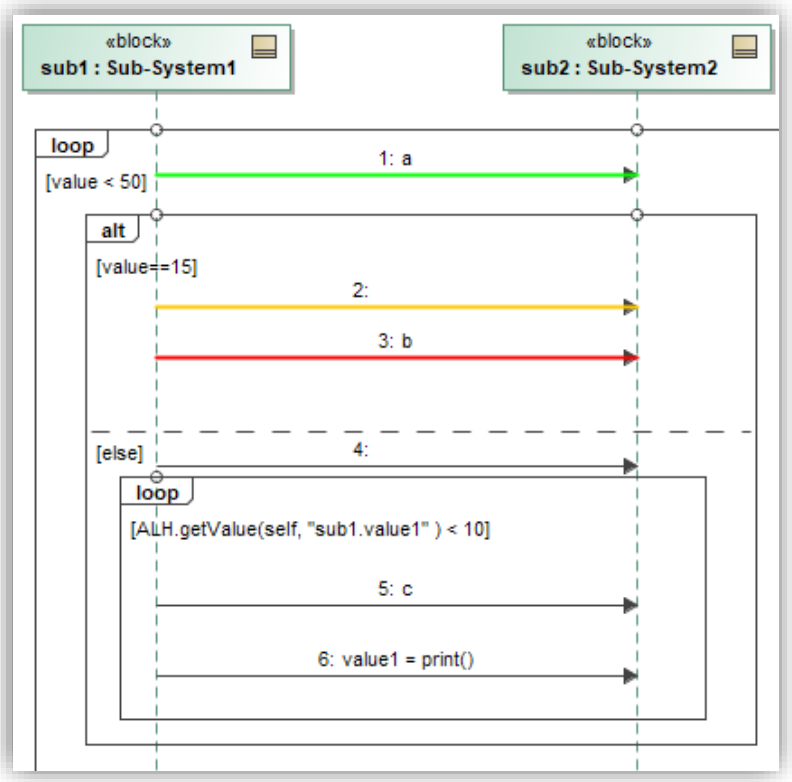
- Interaction Operands
  - Regions inside the combined fragment rectangle… separated by dashed line can be 2 or more
- Each Interaction Operand can have a guard to allow or deny its operation
  - Shown by boolean on the lifeline inside brackets - [ ]
  - Sometimes modeling tools don't follow the sysml rule: Magic Draw shows the guard beneath the header of the operand

OPT – Combined Fragment

- OPT combined fragments will have
  - Exactly 1 interaction operand (cannot be multiples)
  - Signifies that the combined fragment represents a choice of behavior where either the (sole) operand happens or nothing happens
  - Always has a guard…
    - If true, then all events occur
    - If false all skipped



ALT – Combined Fragment

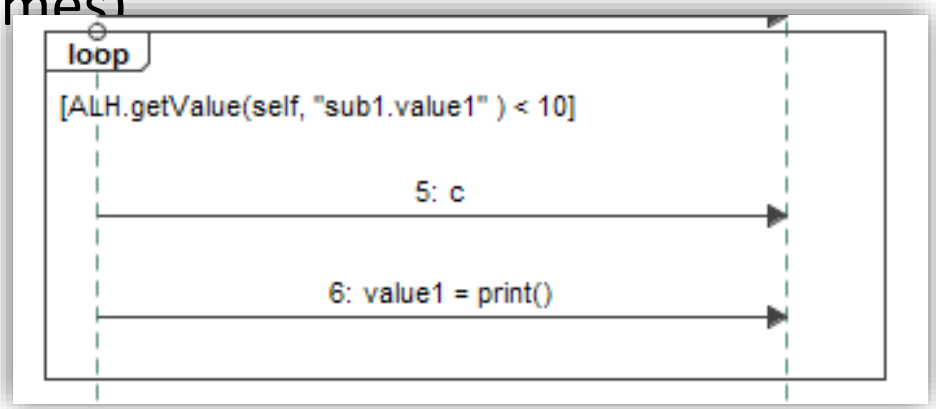


- ALT combined fragments will generally have 2 or more interaction operands:
  - If then, else – If one of the guards is true, then that operand is executed
  - ALT combined fragments do not require an [else] guard on the diagram
- Multiple operands cannot be true… only one

LOOP – Combined Fragment

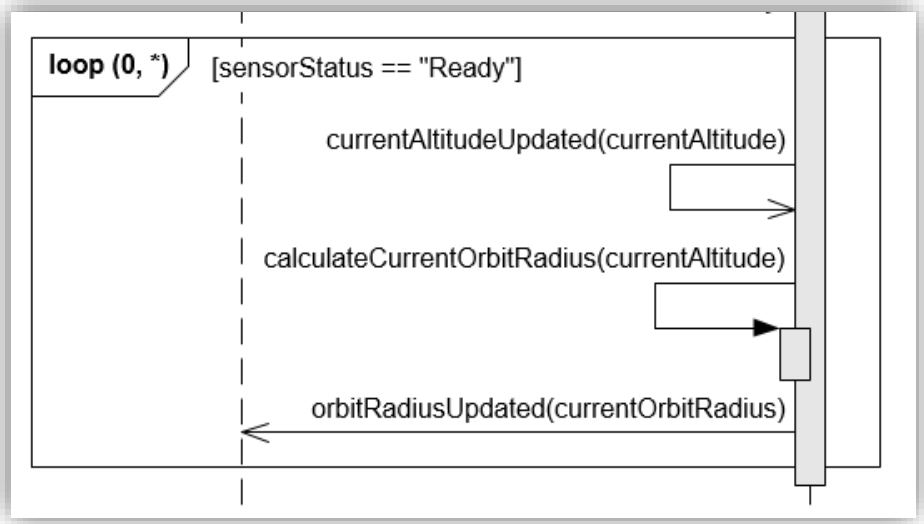
- Will have exactly one region, and MAY have a guard, but is not required
  - Loop will iterate to the upper limit and then end

- LOOP encapsulates a subset of event occurrences that could occur iteratively (multiple times)

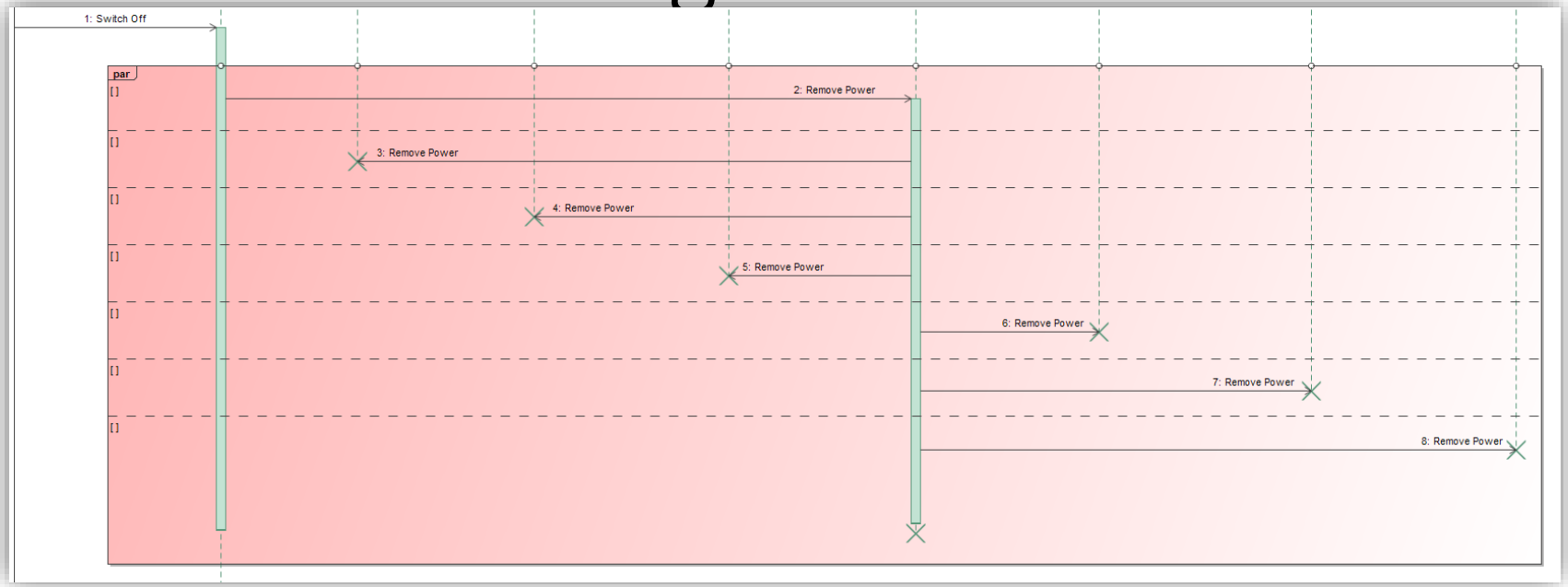


LOOP – Combined Fragment

- In the LOOP header, the parenthesis contains the lower and upper bound for min and max iterations
  - 0 loops, 1 or more, or * for unlimited
- Iterations are not explicit, but show a limit of what's possible
  - (3,10) = minimum of 3, maximum of 10 (unless the guard is activated)
- The guard only becomes meaningful once you have iterated the minimum (lower bound) of loops
  - The loop will terminate when the guard is false, or the max iterations is reached



PAR – Combined Fragment



- A PAR combined fragment represents subsets of event occurances that appear concurrently (in parallel) with one another
- Within a PAR, the order of event occurances that appear in different operands, is non-determnisitic (any order is valid)
- PAR in plain english: The par combined fragment removes the sequential order of the timeline from the operands

Sequence Diagrams Question 9: Combined Fragments

- What should happen after message X3( ) is received by :System3?

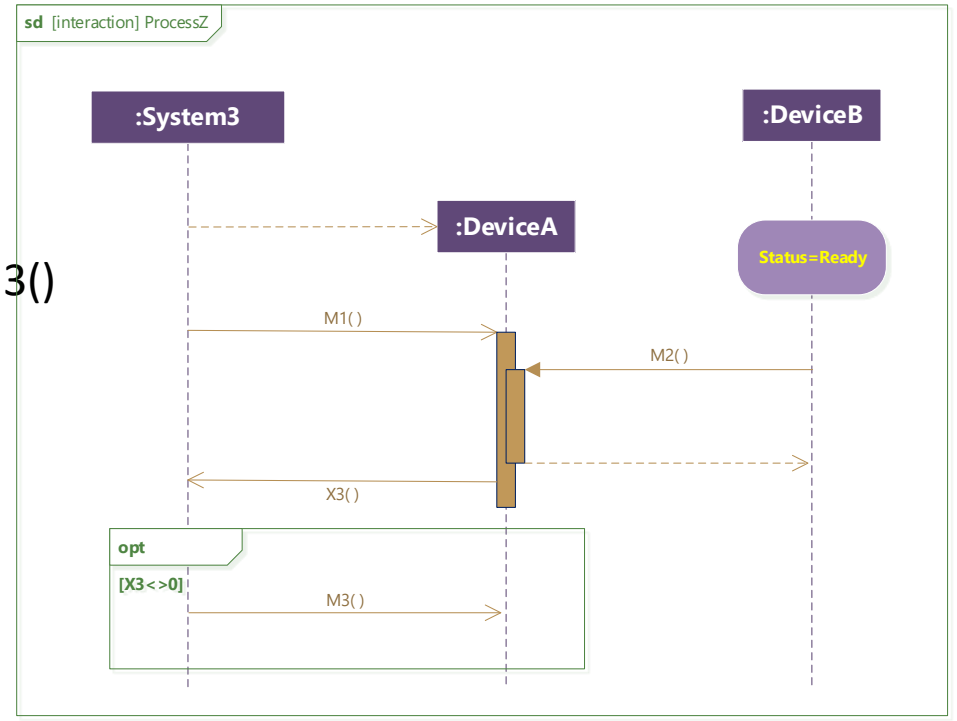
a) The value of X3 is included in message M3()

b) If X3 = 0, then message M3() is sent

c) If X3 <> 0, then message M3() is sent

d) X3 is set to 0 prior to M3() being sent

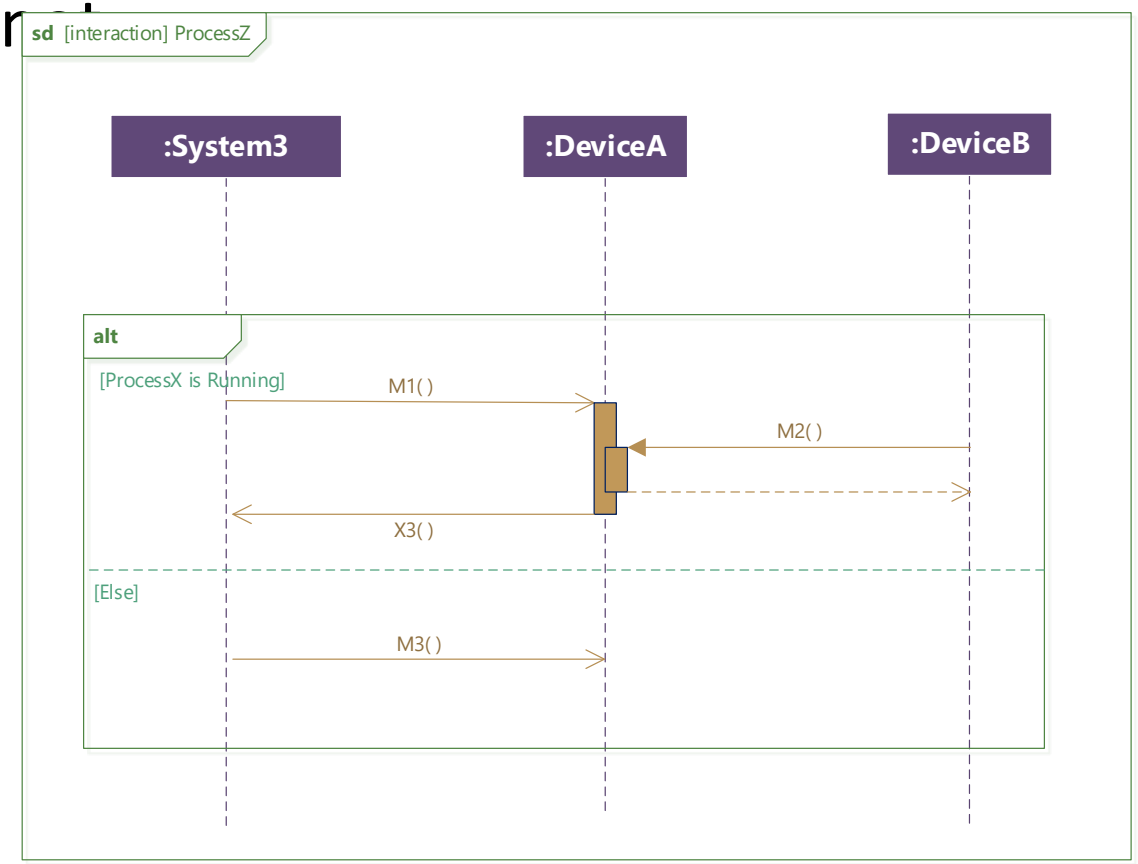
e) None of the above



Sequence Diagrams Question 10: Combined Fragments

- How many event occurances will occur if “ProcessX” is r
  running?

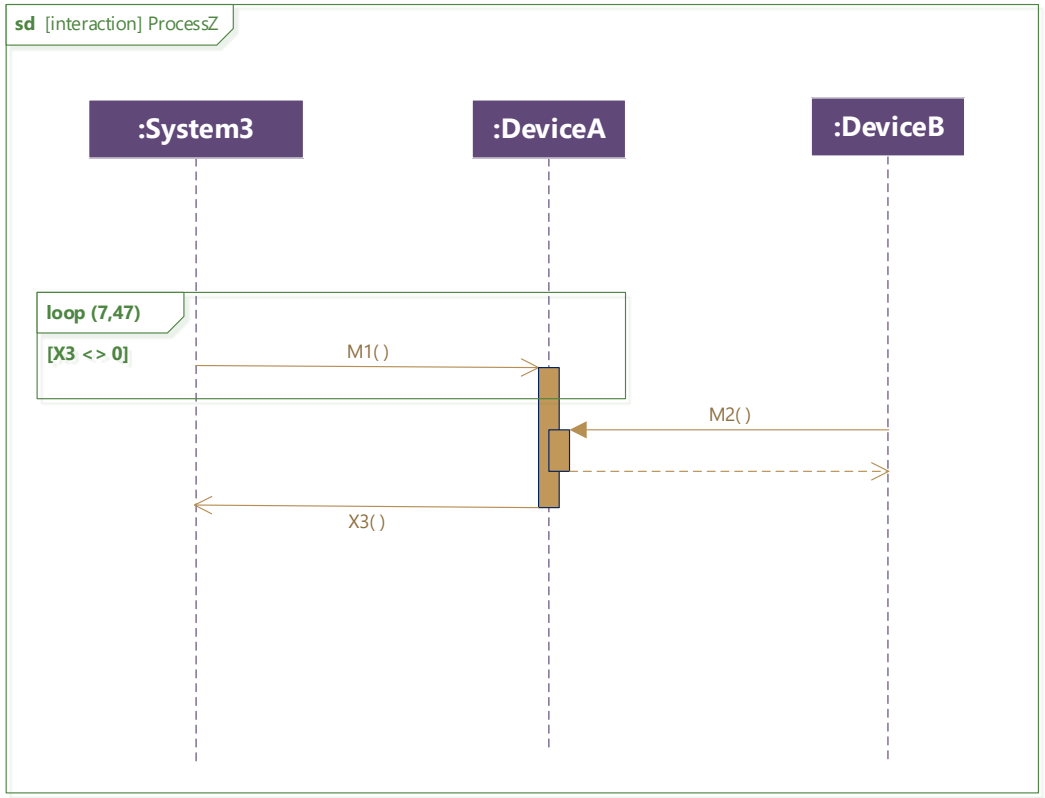
a) 12
b) 8
c) 6
d) 2
e) 0



Sequence Diagrams Question 11: Combined Fragments

- What is the minimum number of times message M1() will be sent to :Device=A?

a) 0
b) 3
c) 7
d) 47
e) Unable to determine



Sequence Diagrams Question 12: Combined Fragments

- Which device is the last to receive power?

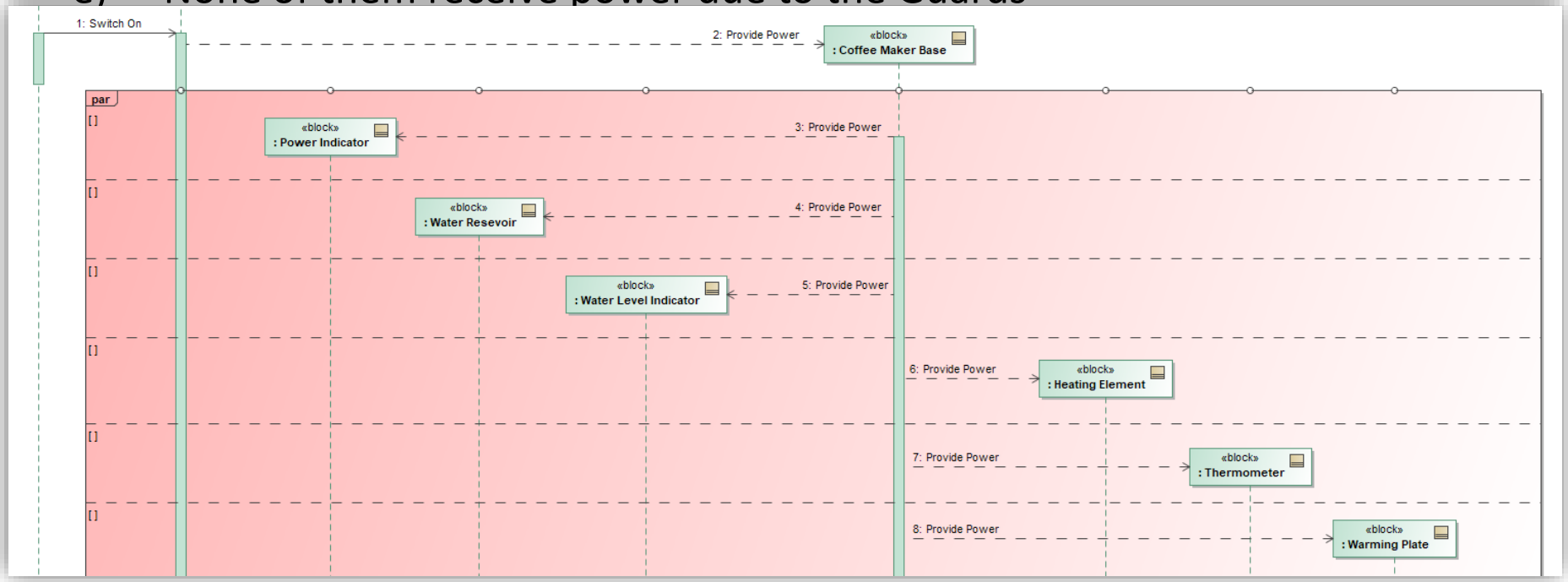
a) :Warming Plate

b) :Coffee Maker Base

c) :Water Level Indicator

d) All of them receive power at the same time

e) None of them receive power due to the Guards



Constraints

- 3 possible ways to use “Constraints” on a sequence diagram…
  - Time Constraint: Specific time or window of time
  - Duration: Length of time
  - State Invariant: Boolean state must be true

Time Constraints

- A time constraint is associated with a single event occurance
  - A specific time –or-
  - A window of time during which an event occurrence must happen
- Examples of time constraints…
  - Event time {command.executionTime}
  - Specific time {2:00 a.m. GMT}
  - Range - {2:00a.m. GMT.. 2:05a.m. GMT}
- Time Constraints can be applied to any event occurrence and are not limited to start events
- Time constraints ARE limited to a single event occurrence (specific time, or range of time, but not a duration)

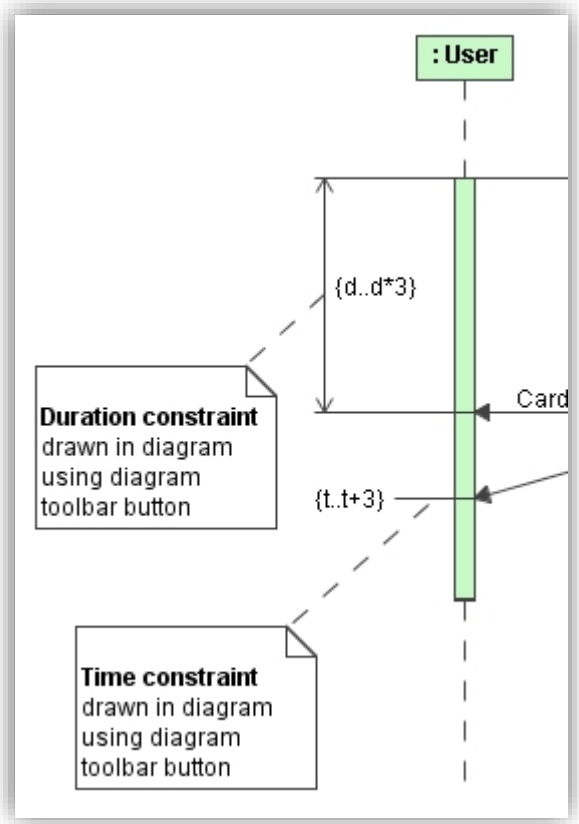


Image courtesy of NoMagic…
https://www.nomagic.com/component/content/category/85-new-and-noteworthy-sections?layout=blog&start=40

Duration Constraints

- Associated with a pair of event occurrences, not a single event
- Can be any associated pair of events

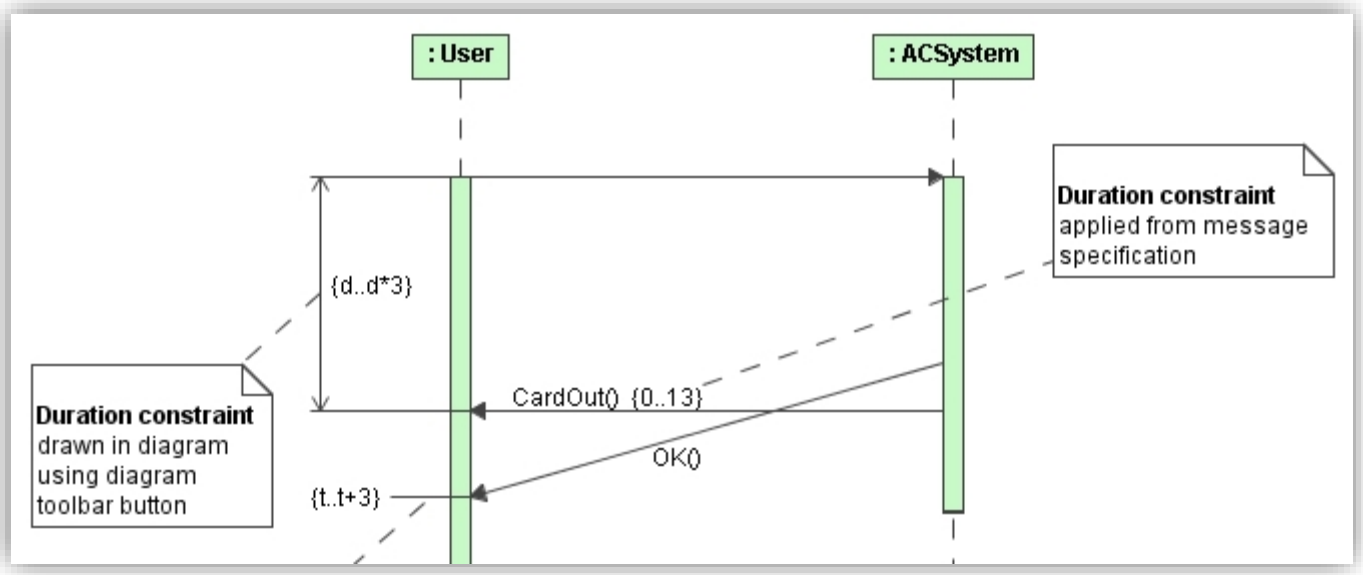
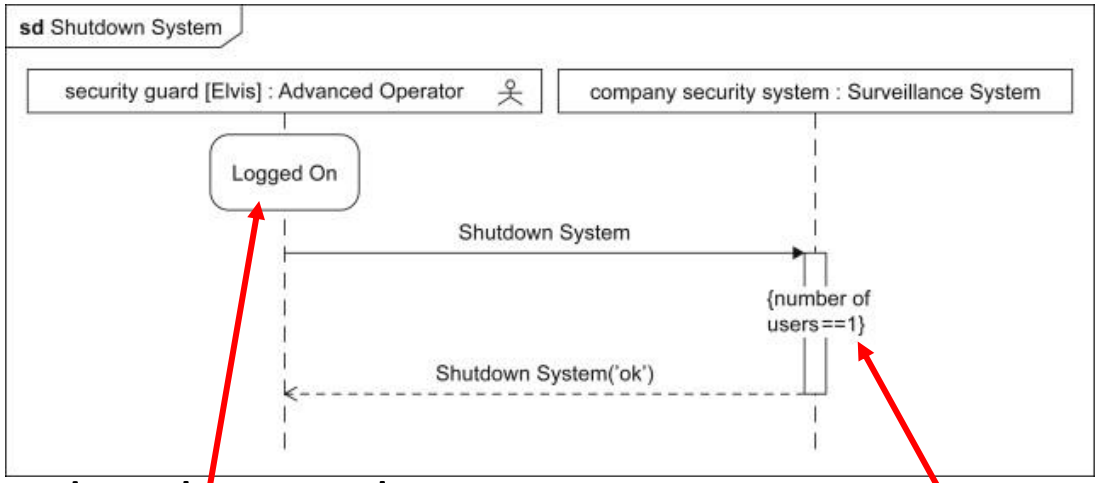


Image courtesy of NoMagic…
https://www.nomagic.com/component/content/category/85-new-and-noteworthy-sections?layout=blog&start=40

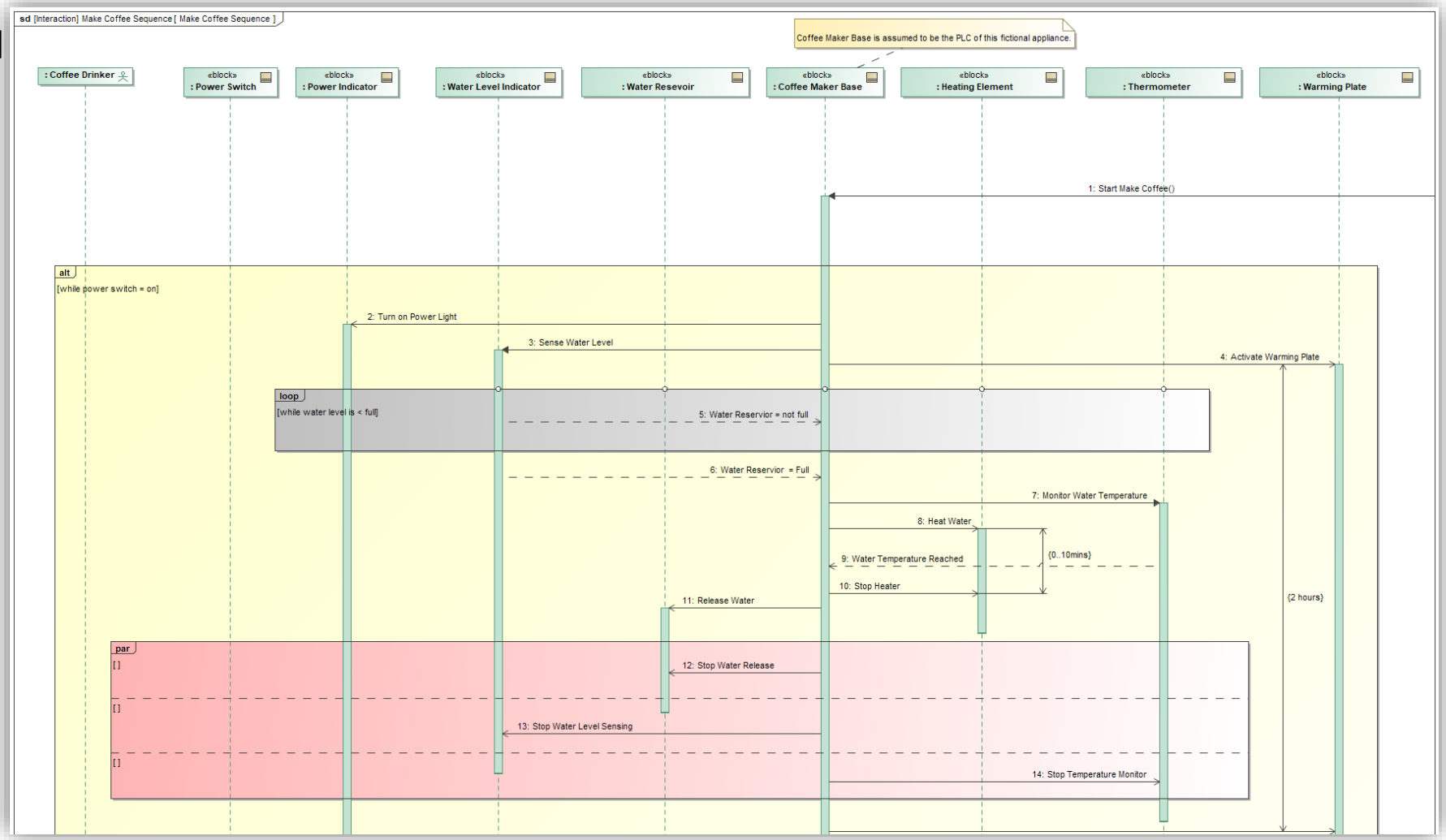
State Invariant Constraints



- Associated with a single event occurrence
- Shown directly above the event on the lifeline:
  - boolean expression inside { }, sitting on a lifeline above the affected event
- The state invariant boolean MUST BE TRUE at the time of the event occurrence, otherwise it the execution of the interaction is deemed invalid
- Alternative representation to represent state invariant…
  - "State symbol" (rounded corner rectangle on a lifeline, represents a state machine elsewhere in the model, named same as a state)

Valid Order of Occurrences

- Shou



Sequence Diagrams Question 7: Constraints

- Based on the diagram provided, Choose the correct list…

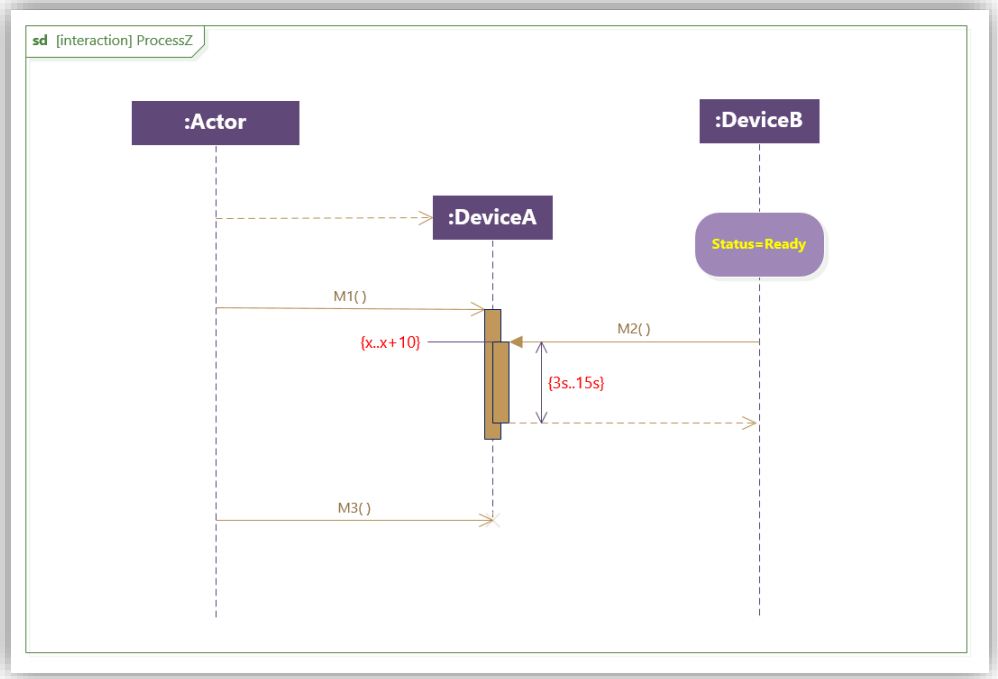
a) “Status = Ready” is a time constraint
“{x..x+10}” is a state invariant
“{3s..15s}” is a duration constraint

b) “Status = Ready” is a state invariant
“{x..x+10}” is a duration constraint
“{3s..15s}” is a time constraint

c) “Status = Ready” is a state invariant
“{x..x+10}” is a time constraint
“{3s..15s}” is a duration constraint

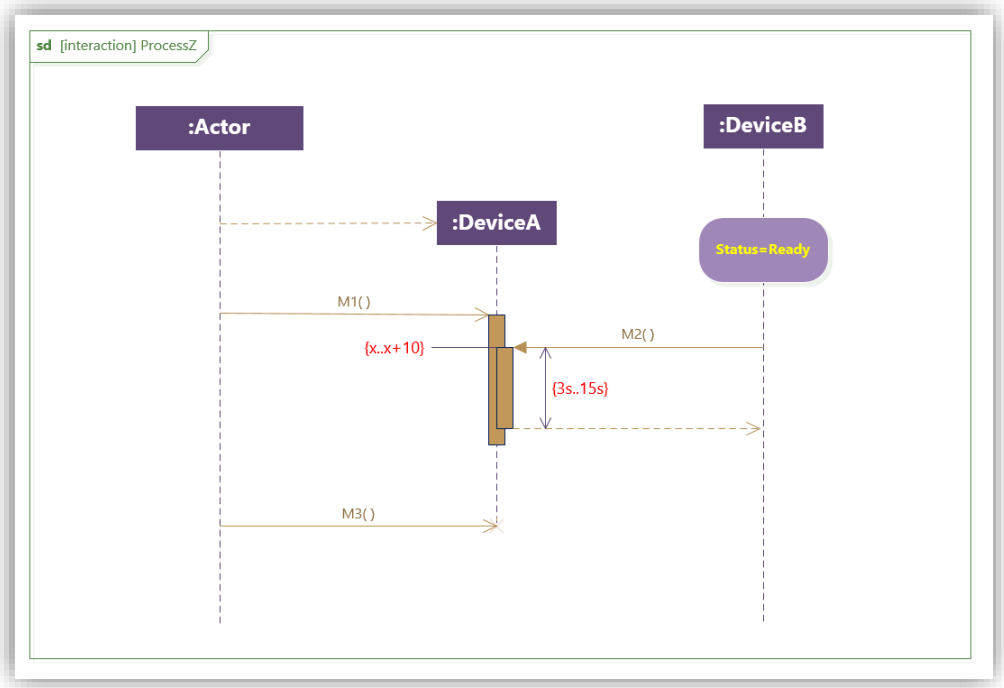
d) “Status = Ready” is a state invariant
“{x..x+10}” is a duration constraint
“{3s..15s}” is a duration constraint

e) None of these are correct



Sequence Diagrams Question 8: Constraints

- Based on the diagram provided, Choose the correct option…



a) Message M2() must be received between time “x” and “x+10” for the execution specification to begin

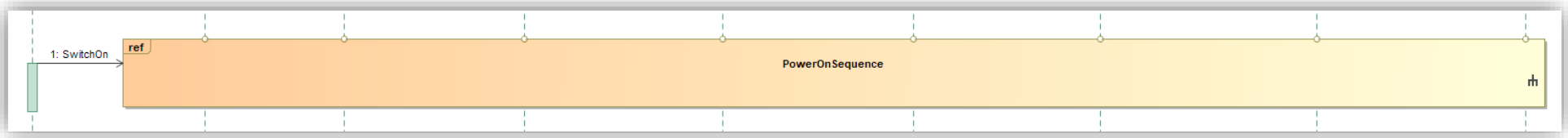
b) The reply message for M2() must occur at least 3 seconds after M2() message is sent, and no more than 15 seconds after

c) Message M2() cannot be sent if :DeviceB Status is not Ready

d) All of these are correct

e) None of these are correct

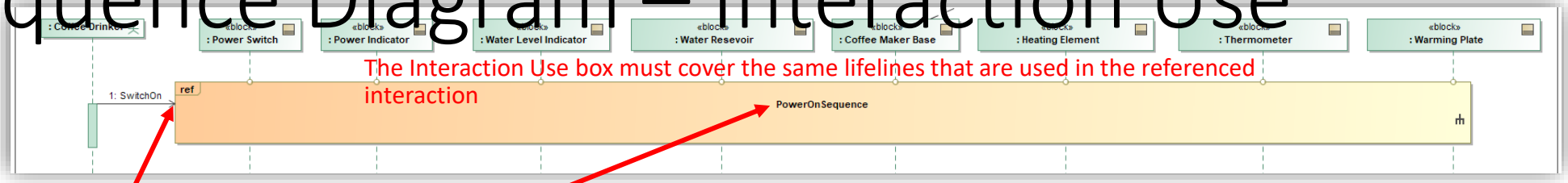
Interaction Use



- Interaction Use appears as a rectangle with a header in upper left corner containing REF abbreviation

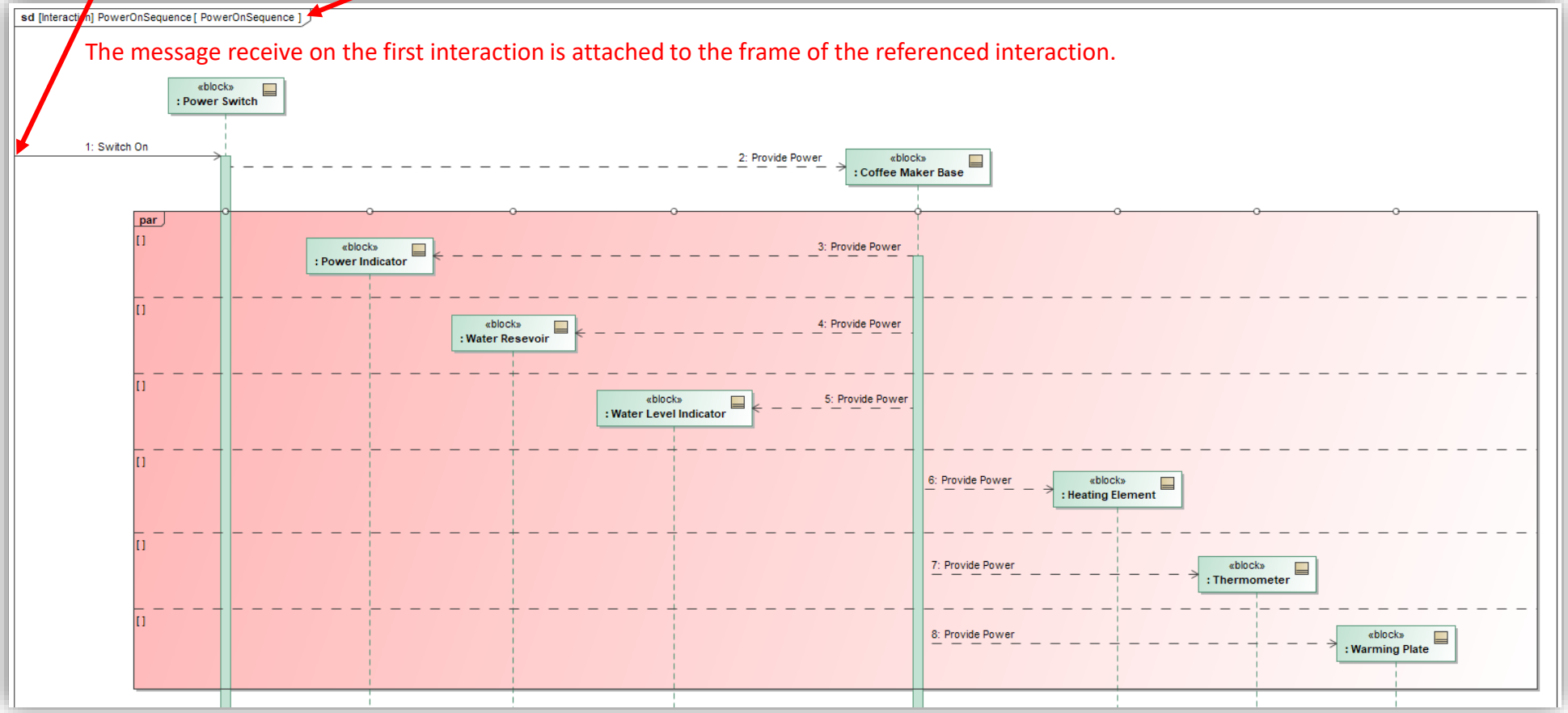
- Interaction use represents a reference to, (invocation of) another interaction elsewhere in the model (one sequence diagram calling another sequence diagram (interaction))

Sequence Diagram – Interaction Use



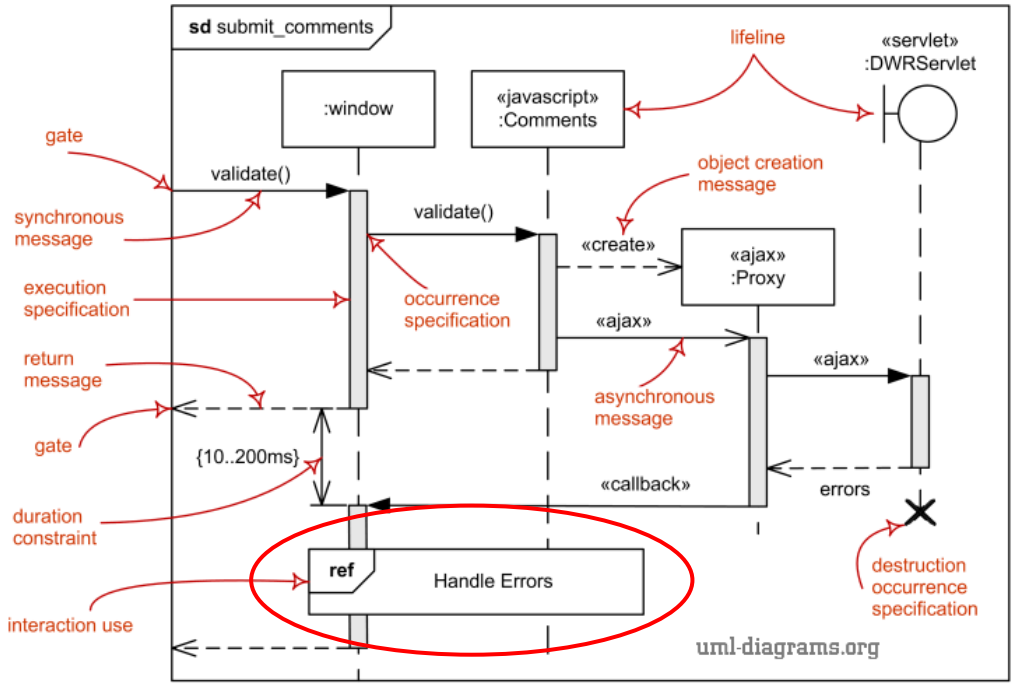
The Interaction Use box must cover the same lifelines that are used in the referenced interaction

…the title of the REF element must be the same as the diagram being referenced



The message receive on the first interaction is attached to the frame of the referenced interaction.

Interaction Use

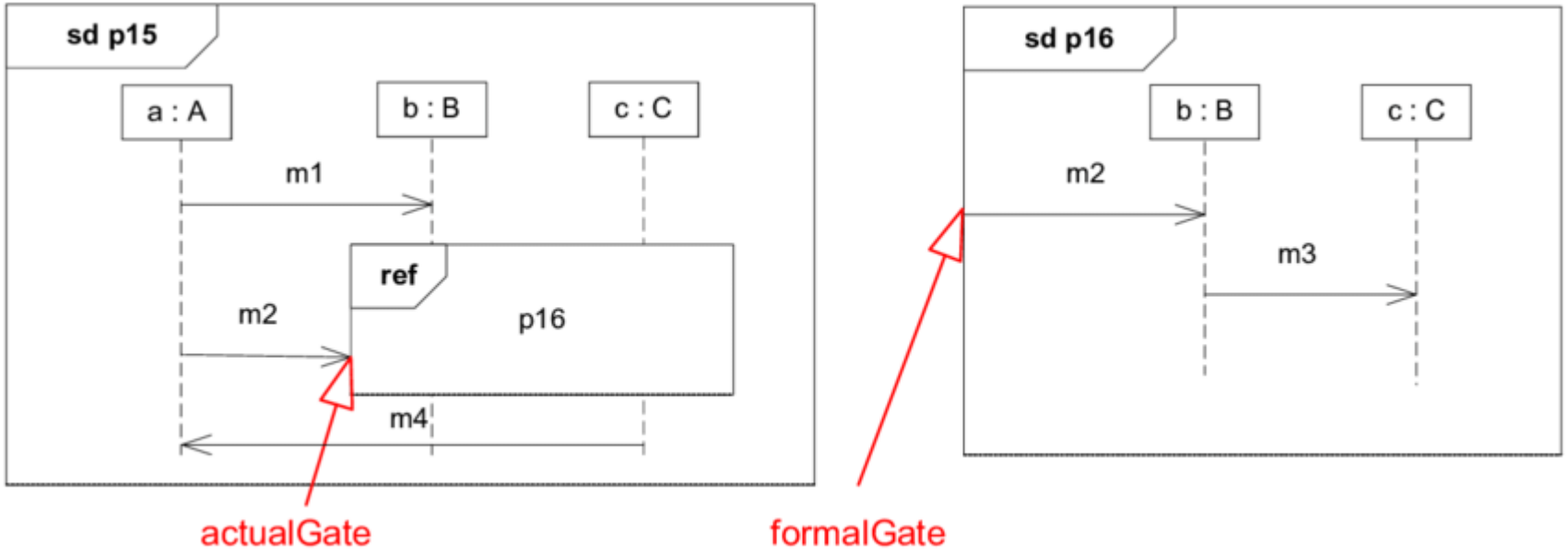


- Interaction use enables the "reuse" of a lower level behavior ("Enables behavioral decomposition")
- Lifeline behind an interaction use conveys involvement, lifeline on top conveys non-ineractive

Image courtesy of uml-diagrams.org

Actual & Formal Gates

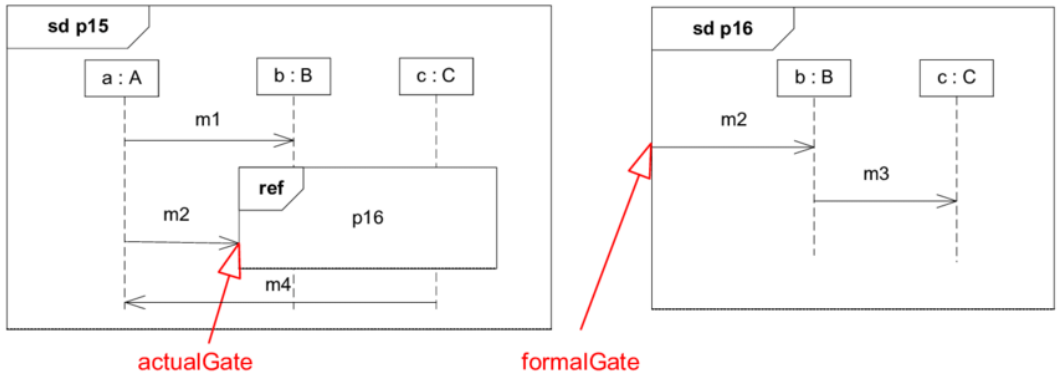
- SysML Jargon:
  - A message enters and leaves an interaction use at "Actual Gates“
    - There is no notation, it is implicit
  - A message enters or leaves an interaction at "Formal Gates“
    - These are the points of entry ON the frame of the diagram.



Gates_fig9_267240554

Actual & Formal Gates

- Interaction uses
  - Quantity of actual versus formal gates must be 1 to 1 for each
    - Same number of actuals in the calling interaction as formals in the called interaction
- Actual vs Formal Pnemonic Device:
  - FF = Formal/Frame
  - Actual on reference frame

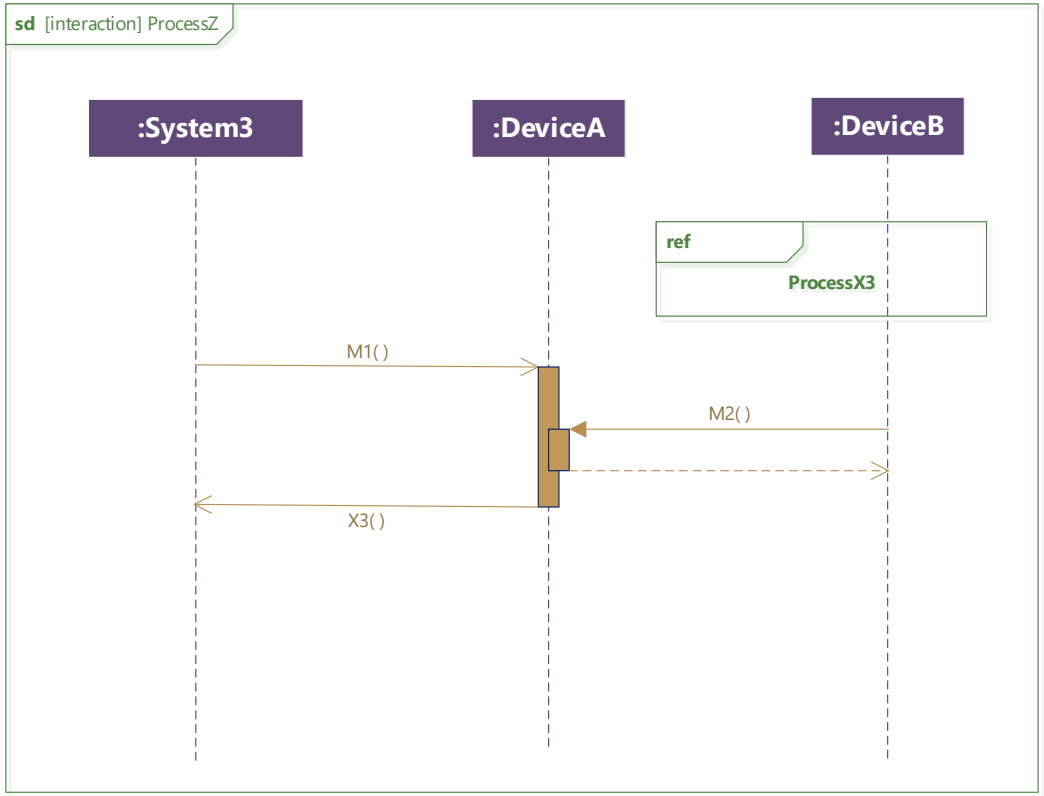


https://www.researchgate.net/figure/Formal-and-actual-Gates_fig9_267240554

Sequence Diagrams Question 13: Interaction Uses

- Which event must occur first?

a) M1() Send
b) M1() Receive
c) ProcessX3
d) M2() Send
e) M2() Receive
f) Unable to determine from this diagram



Sequence Diagrams Question 14: Gates

- How many Actual Gates are shown on this diagram?

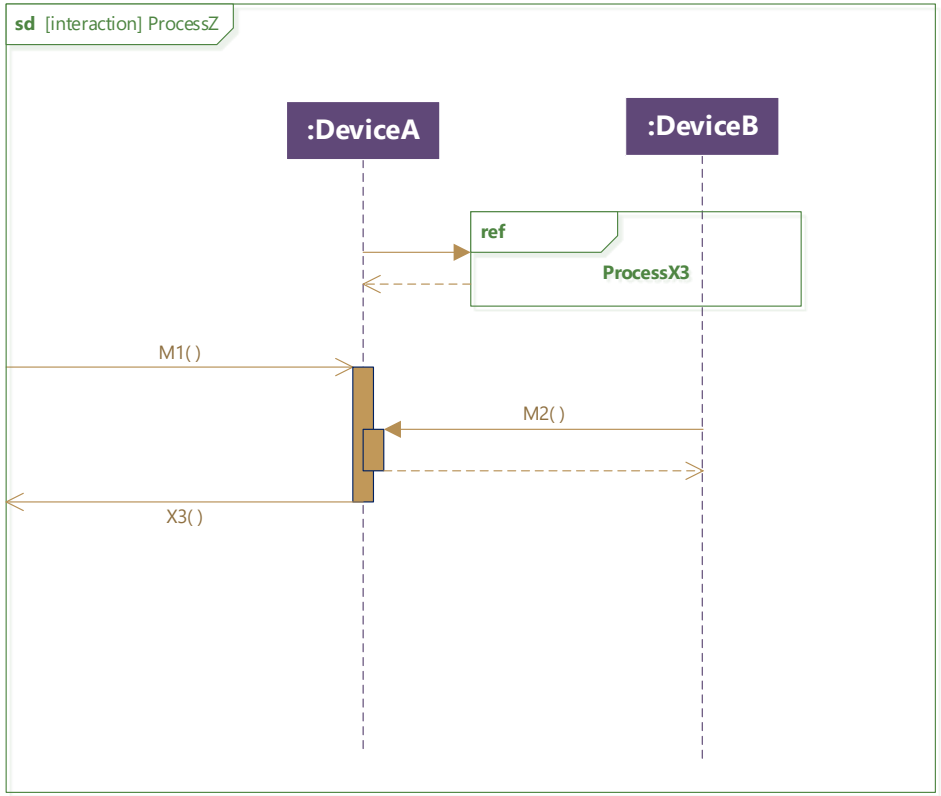
a) 0

b) 2

c) 4

d) 8

e) 16



Purpose and Use of Object Nodes

- 3 kinds of object nodes:
  - Basic object node
  - Pin
  - Activity Parameter

Questions

COLLEGE of CHARLESTON
1770

Summary

References

- Additional information can be obtained by reviewing:
  - SysML Distilled (Delligatti)
    - Chapter 5: Use Case Diagrams