FROM RESEARCH TO INDUSTRY

cea tech

papyrus

www.eclipse.org/papyrus

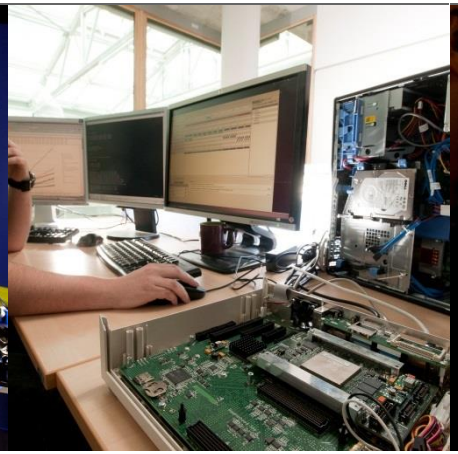
list

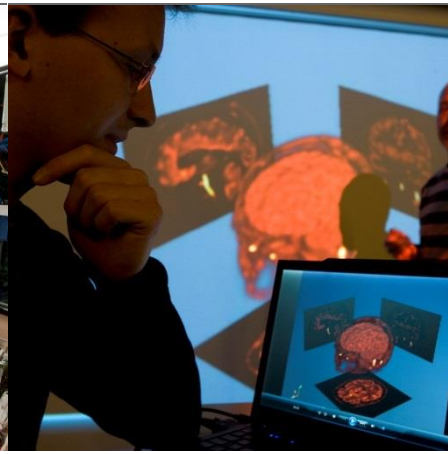
UML BEHAVIORAL STATE MACHINE

Introduction on UML for Industrial Systems

Jérémie Tatibouët, Shuai Li, François Terrier, Sébastien Gérard, **Asma Smaoui**

{first_name}.{last_name}@cea.fr





INSTITUT CARNOT CEA LIST

digiteo

CONTENT

Introduction

- Where does UML state-machines come from ?
- Usages of UML state-machines within a model

Active classes and events

- Relation between structure and behaviors

Basic elements to describe behavior state-machines

- State, Transition, IntialPseudoState and FinalState
- Build an example with the subset explained in this section

Advanced behavior state-machines

- Pseudo-states and parallelism
- Refine your example to integrate usage of advanced concepts

Behavior state-machines: understand the global dynamic

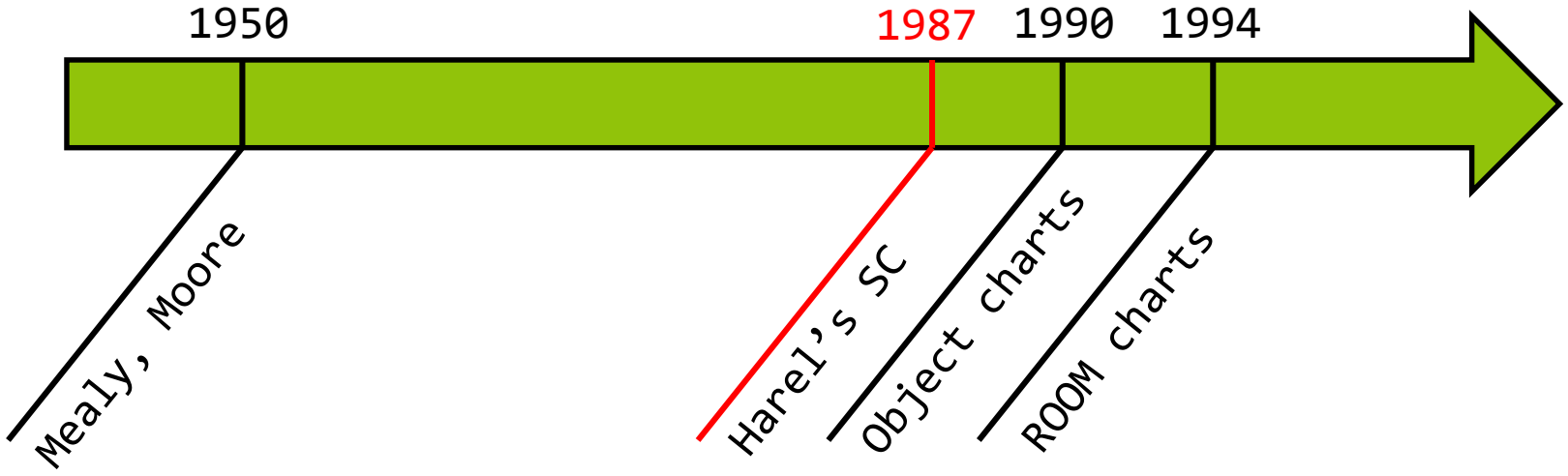
- Event dispatching, Run-to-Completion
- Execute step by step your refined example

Going further: advanced semantics of behavior state-machines

- Transition selection, transition conflict, special kind of transition, deferred event, protocol state-machines

Relation between this course and current research

A BIT OF HISTORY



"The specific form of finite state automata used in UML is based on an object-oriented variant of David Harel's *statecharts* formalism. (However, readers who are familiar with that formalism should note that there is a small number of semantic differences that distinguish the UML version from the original.)"

Excerpt from UML 2.5

Reference	Link
Mealy	here
Harel	here
Object Charts	here
ROOM Charts	here

Based on **Dr. H. Störrle, Dr. A. Knapp** slides

USING STATE-MACHINES IN OUR MODELS

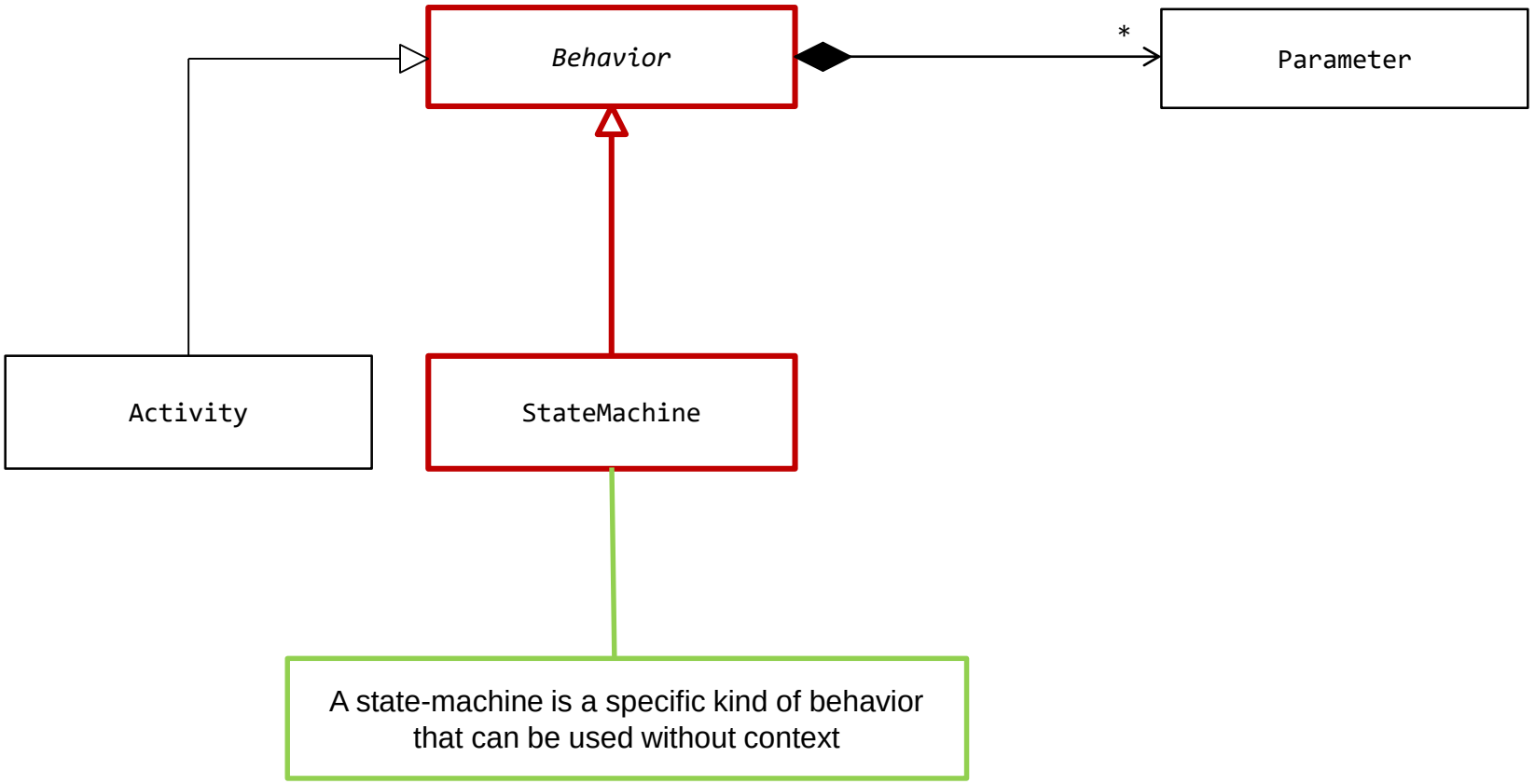
A. Standalone mode
   - A state-machine without context

B. As a method of an operation
   - To describe its implementation
     - Note: this usage is rare

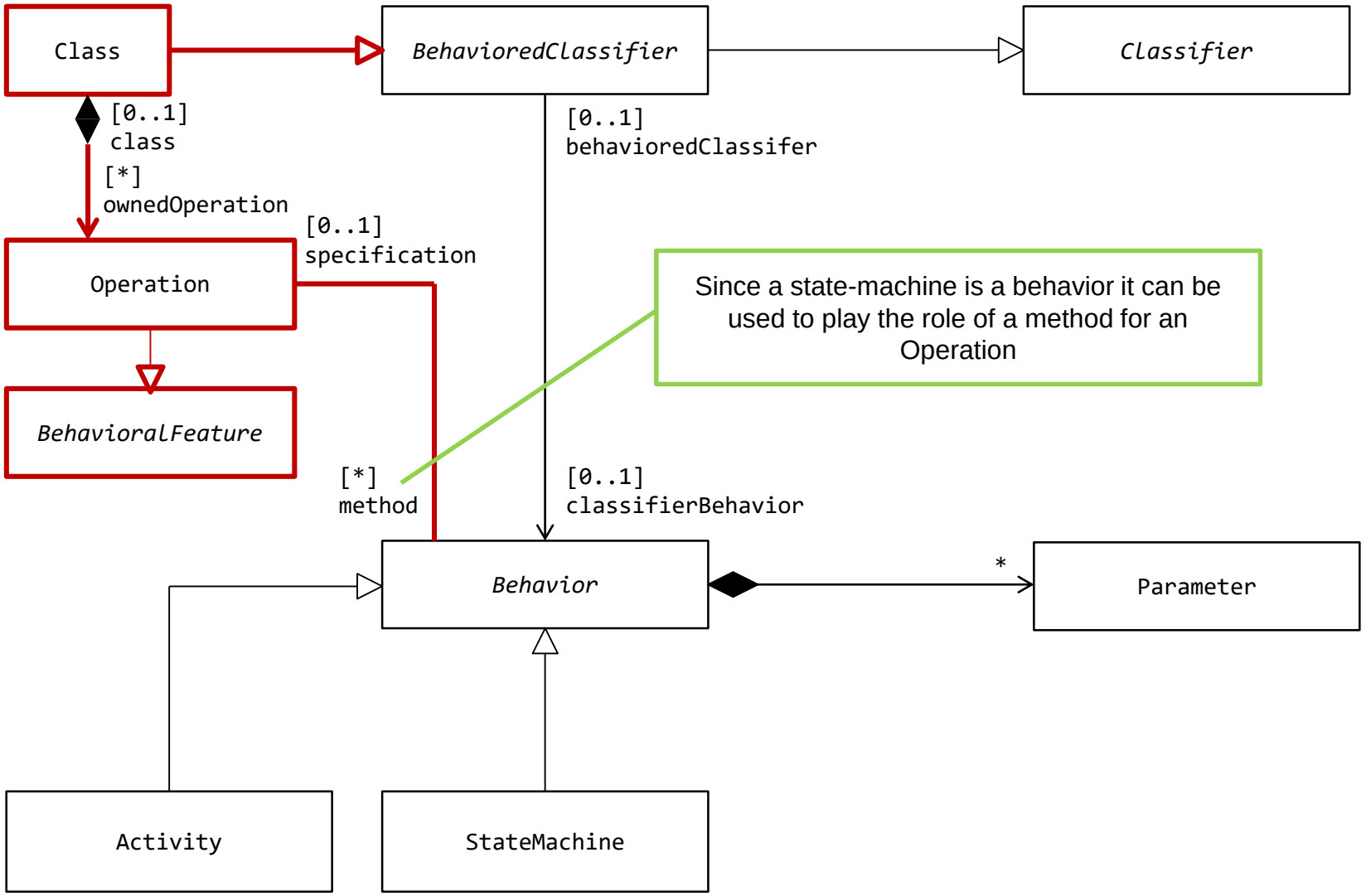
C. As a classifier behavior
   - Used to describe the life-cycle of an object
     - Along this course we will focus on this kind of usage

BEHAVIOR STATE MACHINE

IN STANDALONE MODE

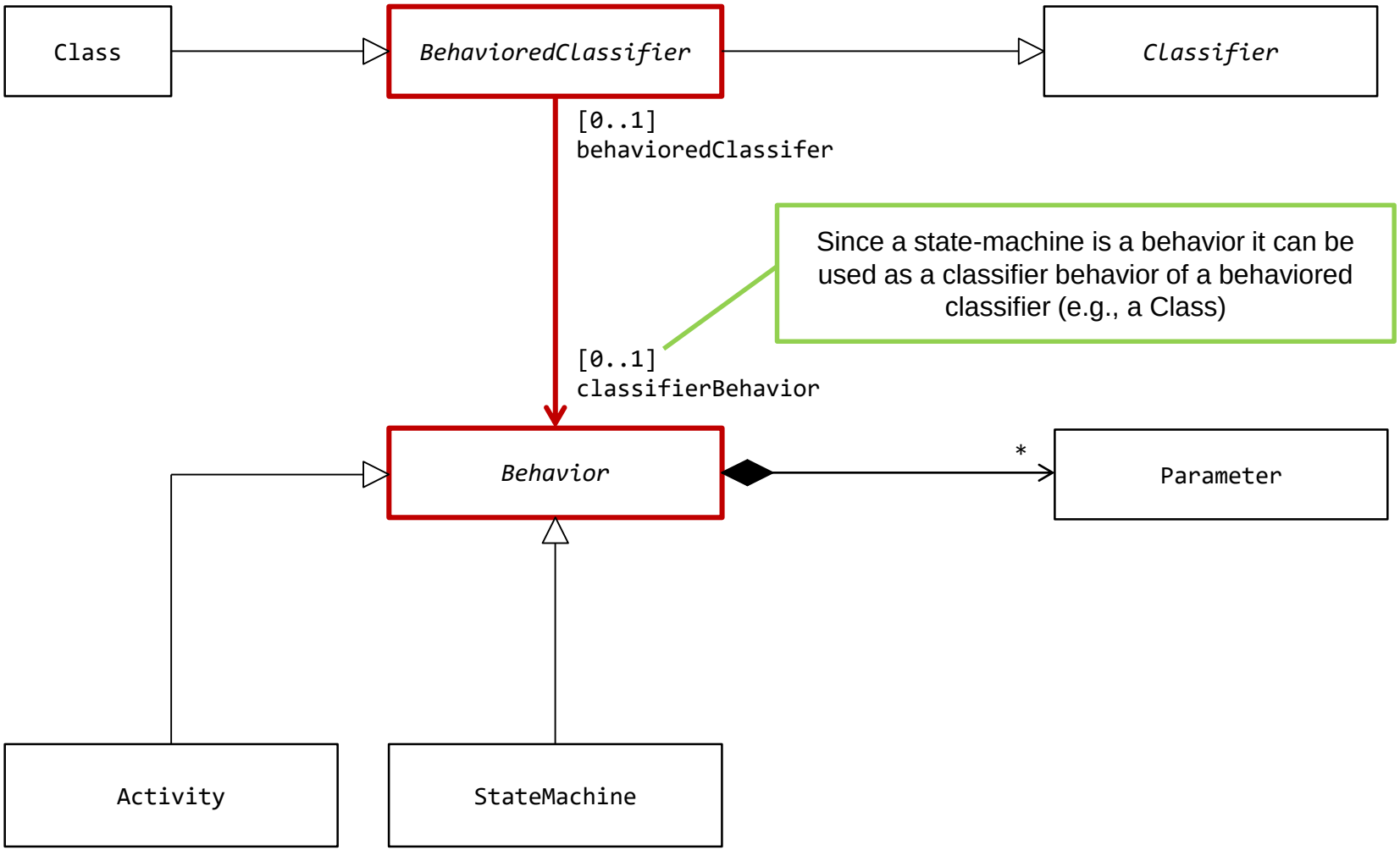


BEHAVIOR STATE MACHINE
AS AN OPERATION IMPLEMENTATION



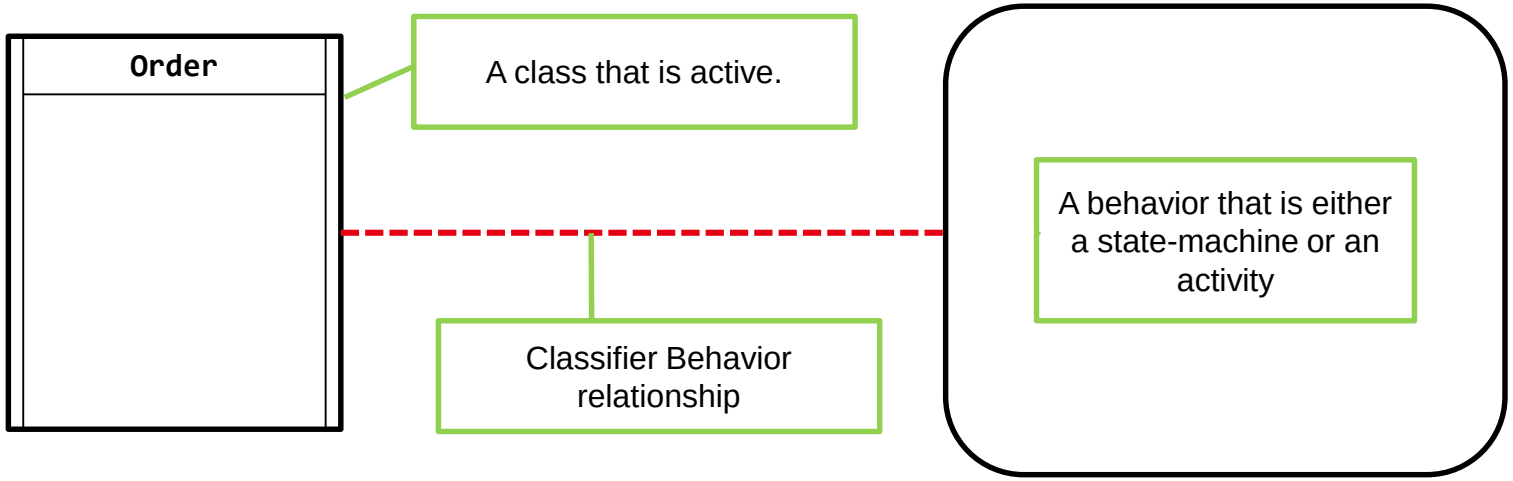
BEHAVIOR STATE MACHINE

AS CLASSIFIER BEHAVIOR



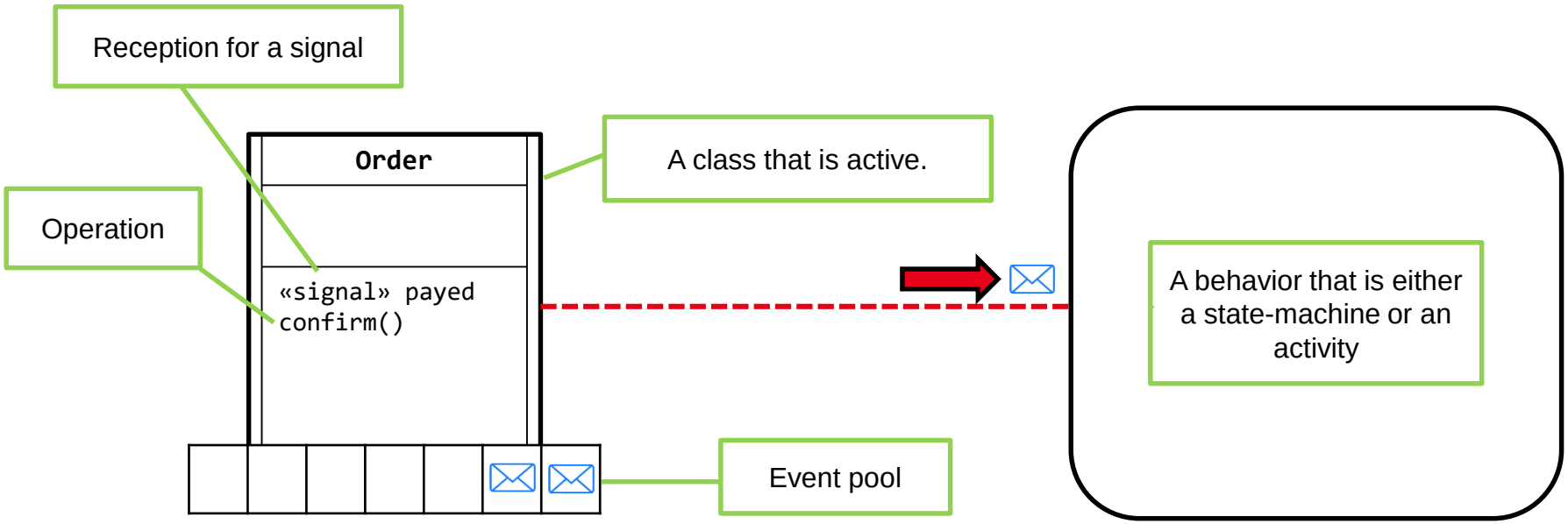
EXAMPLE

ASSOCIATE A CLASS TO ITS CLASSIFIER BEHAVIOR



In papyrus

- Select the class
- In the UML property view of this class set the “isActive” property to true
- Click on the advanced of the property view
- For the “classifierBehavior” line select the behavior that you want to associate with this class

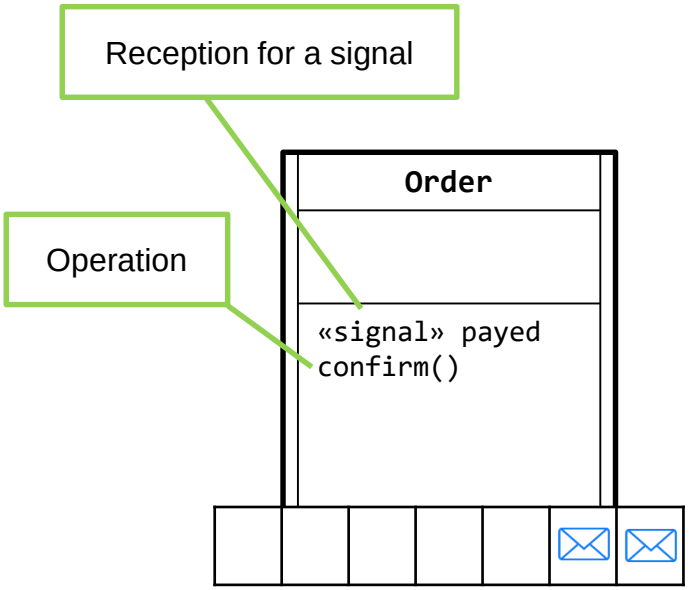


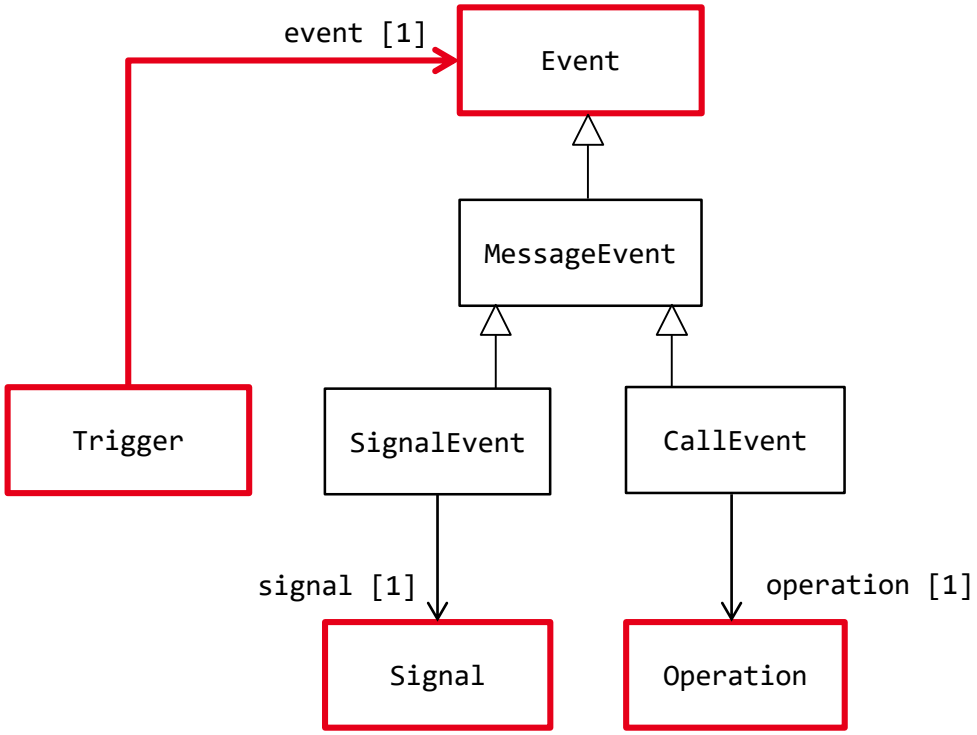
Active classes and event pool

- An active class may receive event occurrences
- These event can be of different types: CallEvent and SignalEvent
- The event occurrences are posted in the event pool of the class
- One event at a time is dispatched on the classifierBehavior
- The dispatching of the event might triggers a set of reactions in the classifierBehavior

TYPE OF EVENTS

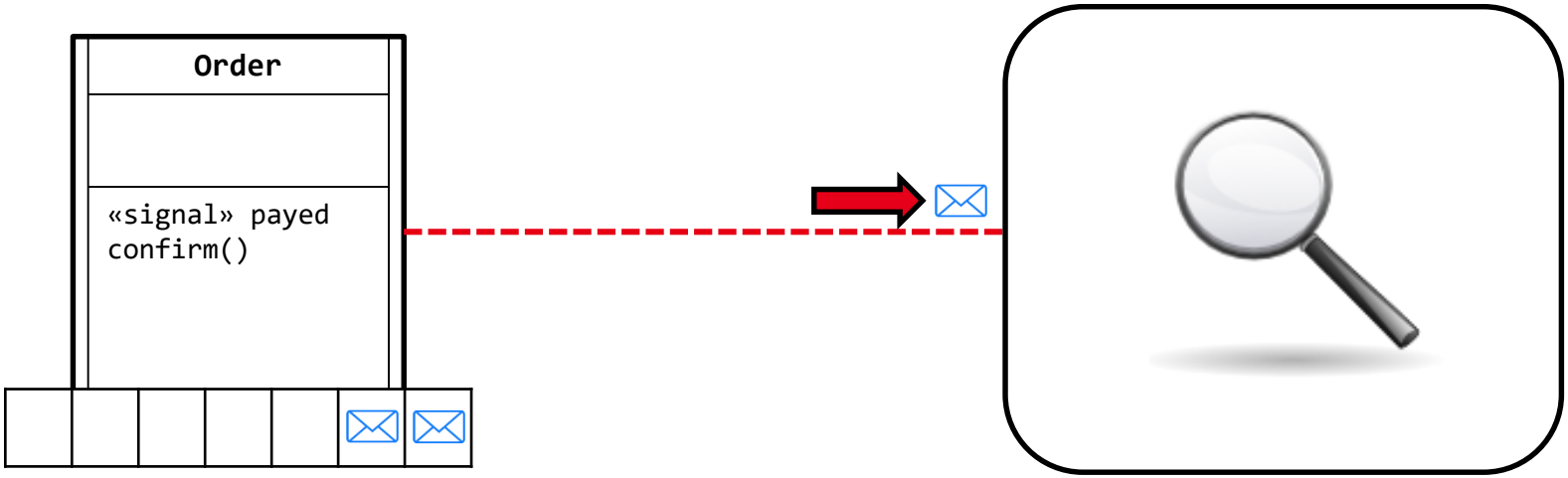
CALLEVENT AND SIGNALEVENT





Event machinery

- Trigger
  - Declared by an element of a behavior
  - It specifies an explicit waiting for a specific type of event
- SignalEvent
  - The expected event is signal
- CallEvent
  - The expected event is an operation call
- Note: there are other kind of events
  - e.g., ChangeEvent, TimeEvent

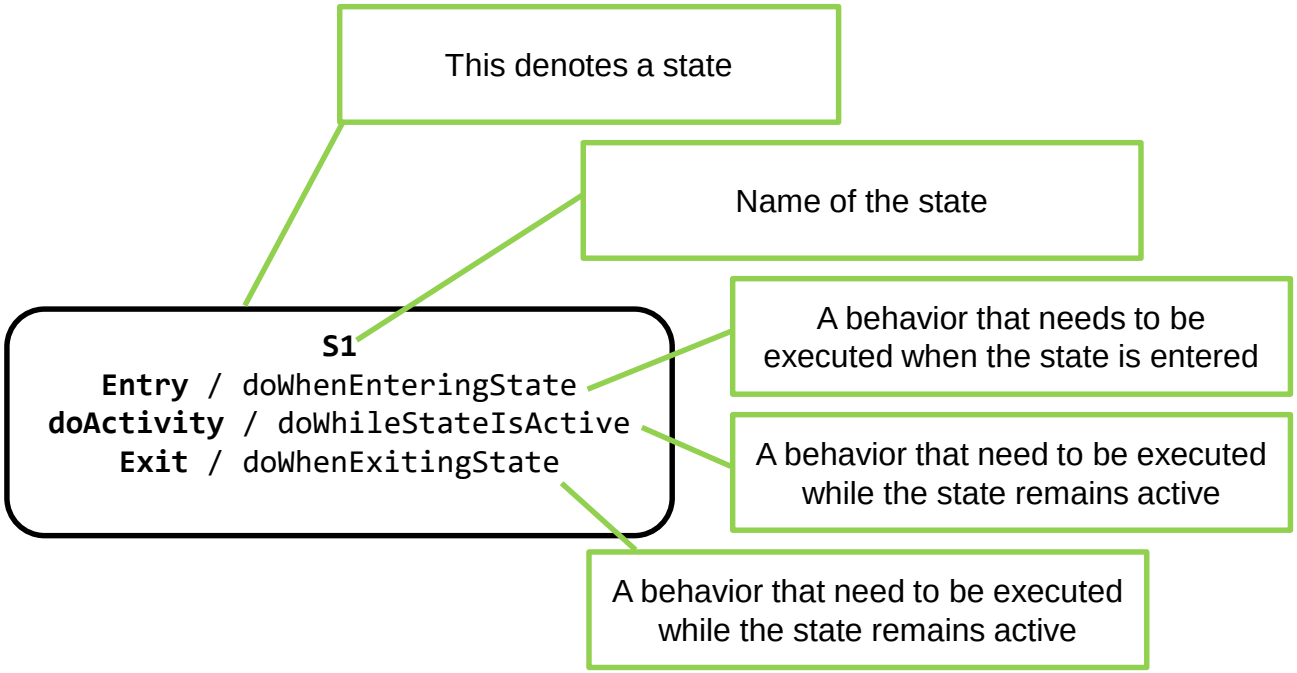


Objectives

- Understand the syntactic constructions required to build state-machines
- Understand the semantics of these constructions

State

- "A state models a situation during which some invariant condition holds"

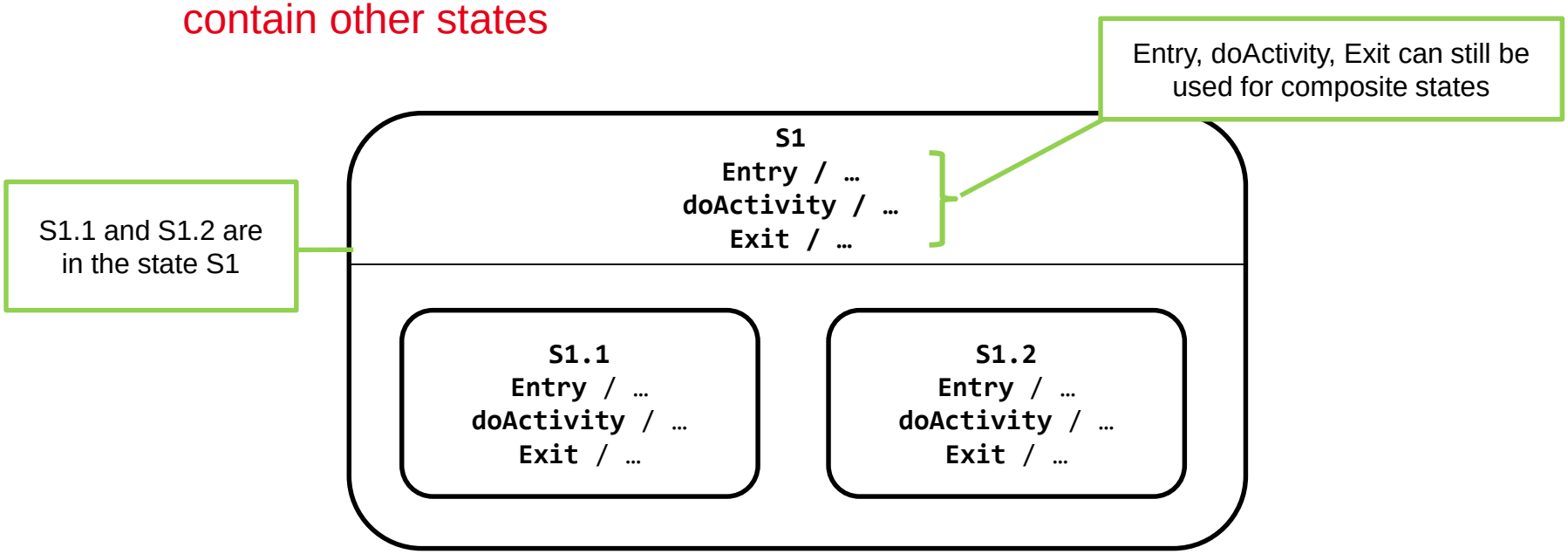


- Example: an order can be Payed, Confirmed, Packed, Cancelled,…

Payment was done and confirmation of the order was sent to the client

State (composite)

- In addition to what is possible with a simple state, a composite state can contain other states

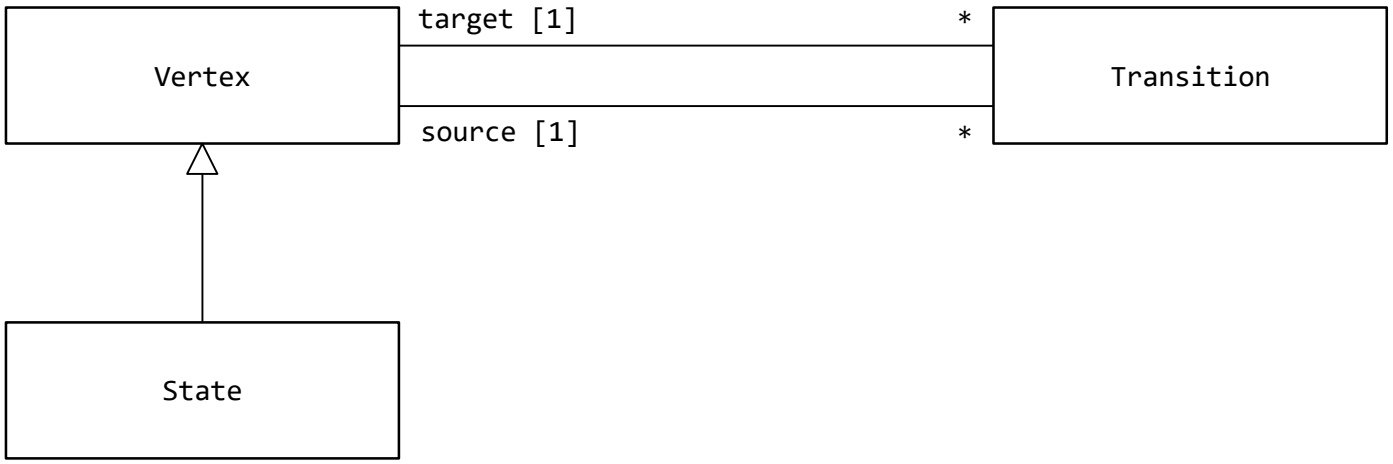


- If S1 is active that means one of the states which is included in S1 is also active (e.g., situation where S1 is active as well as S1.1).

- Example: An order can be in the process of being Finalized. This process usually includes the payment and the sending of a confirmation of the order. Therefore Finalized can contain both Payed and Confirmed.

Until now

- We talked about states (simple and composite)
- What is missing here to complete simple state-machines?
  - How do we move from a source state to a target state ?



Transition

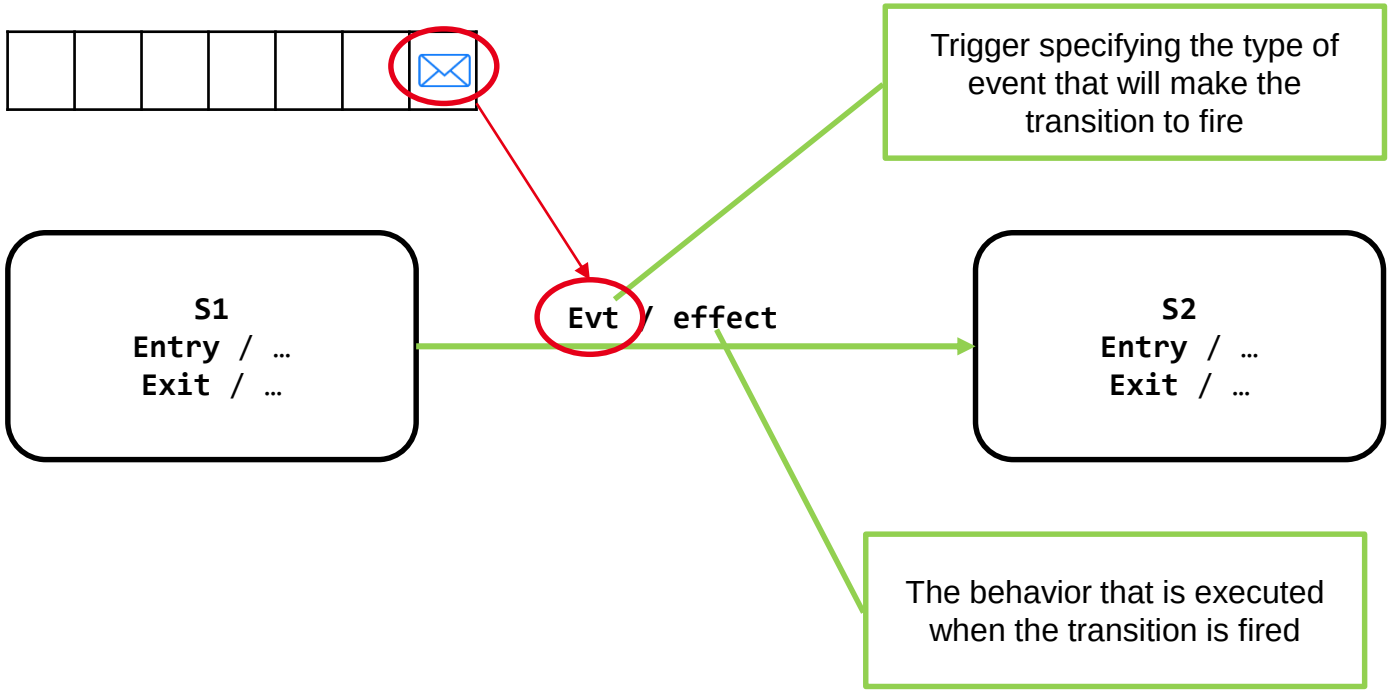
- A transition is an edge between one source vertex and one target vertex
  - Note: both source and target can be the same

- It denotes part of a path that can be followed during the execution of step of a state-machine.

- Completion transition: There is no constraint(s) to fire this transition. It can be fired immediately after S1 was exited.

Transition

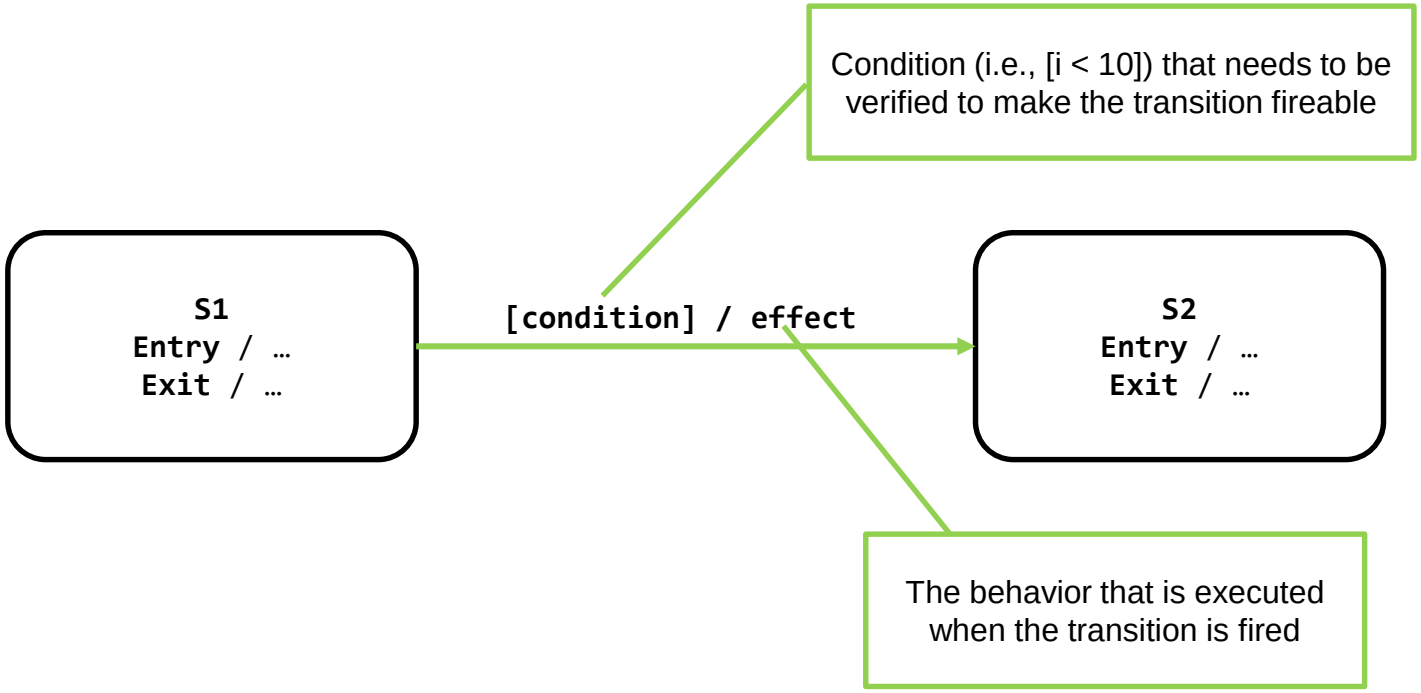
- Can be triggered
  - Fires when a specific event is available



- Note: A single transition can specify multiple triggers (i.e. it can fire upon the arrival of many different events)

Transition

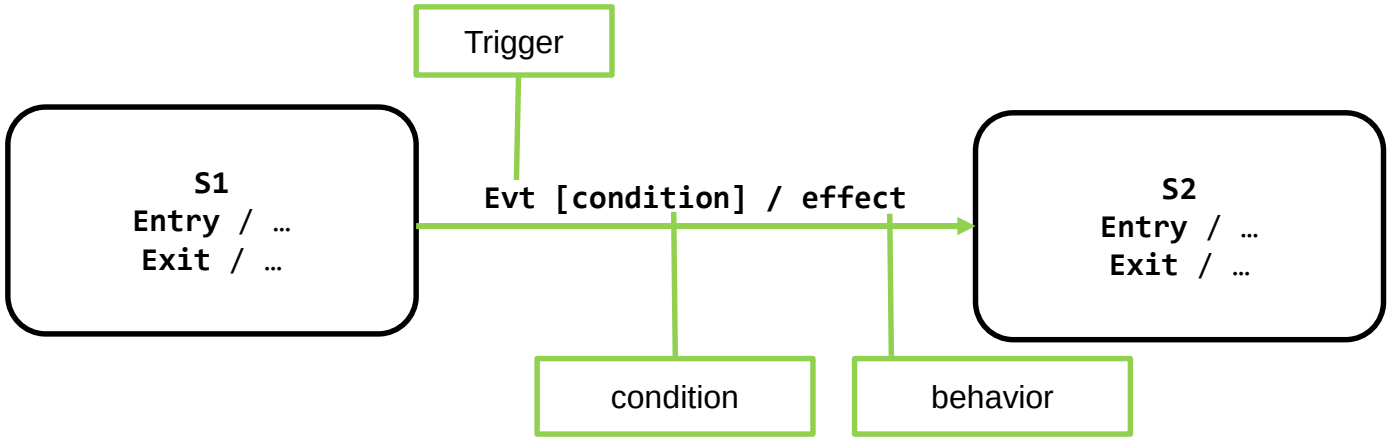
- Can be guarded
  - Fires when a the condition placed on the guard is verified



- Note: In the context guards are specified using UML Constraint
  - Make the specification uselessly complex
  - Should be a lot simpler (e.g., expressions)

Transition

- Can be guarded and triggered
  - Fires when a specific event is available and the condition specified holds

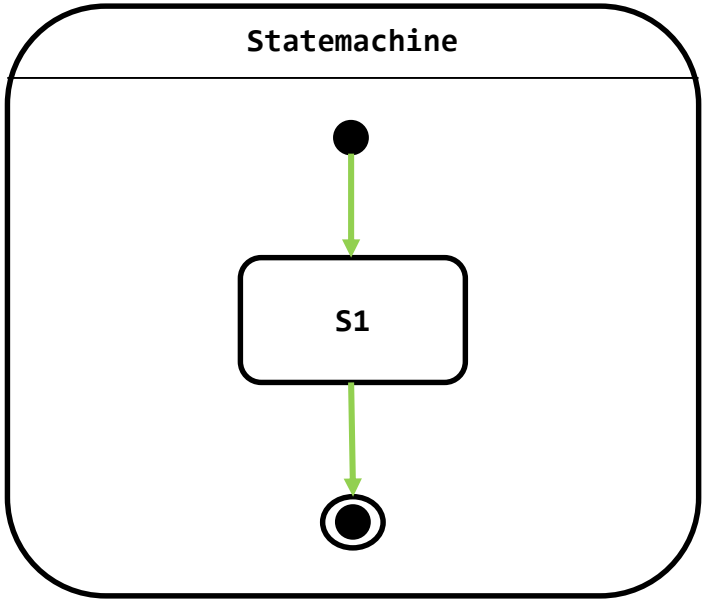


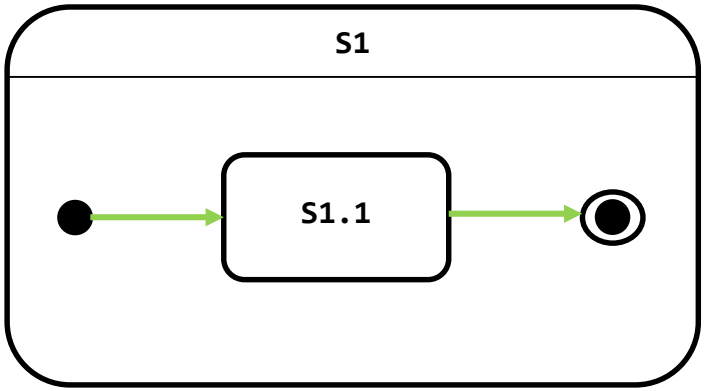
A. Initial pseudo state ●

- Define the starting point of a state-machine
- Define the starting point of a behavior owned by a composite state

B. Final state ◉

- Specialization of a state
- Define the end point of a state-machine
- Define the end point of a set of states owned by a composite state





Drink dispenser specification

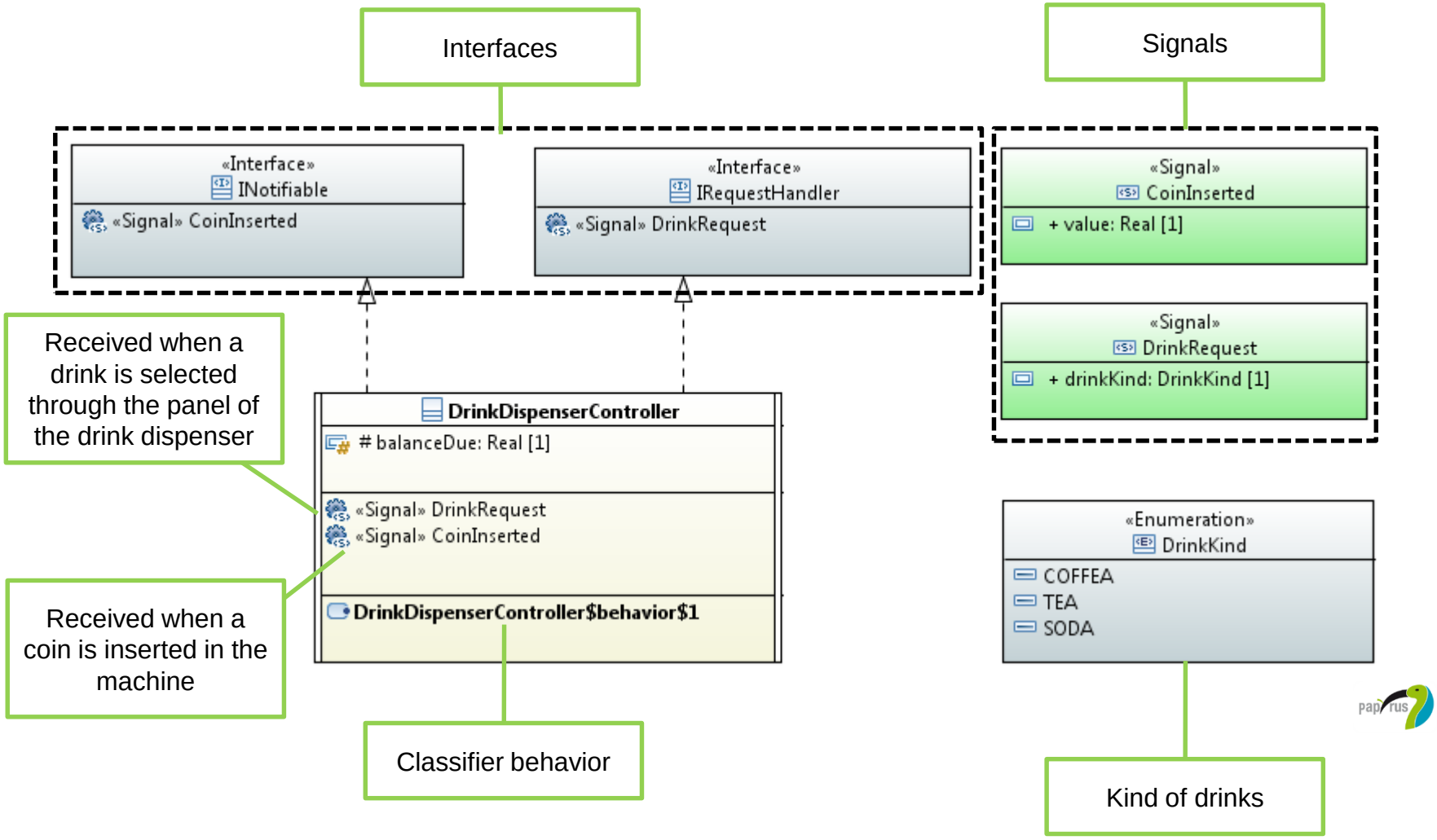
- Wait until a user chooses a drink
- Handle payment with cash
  - Payment is not complete while the cash provided is not equal or greater than the price of the selected drink
- When payment is completed
  - If required, return change
  - Prepare and deliver the drink
- There are three type of drinks
  - Coffee (1€), Tea (1.10€) and Soda (1.30€)

Exercise (40 min)

- Define the architecture of the controller of the machine
- Define the classifier behavior of the controller using a behavior state-machine

BUILD AN EXAMPLE (2/4)

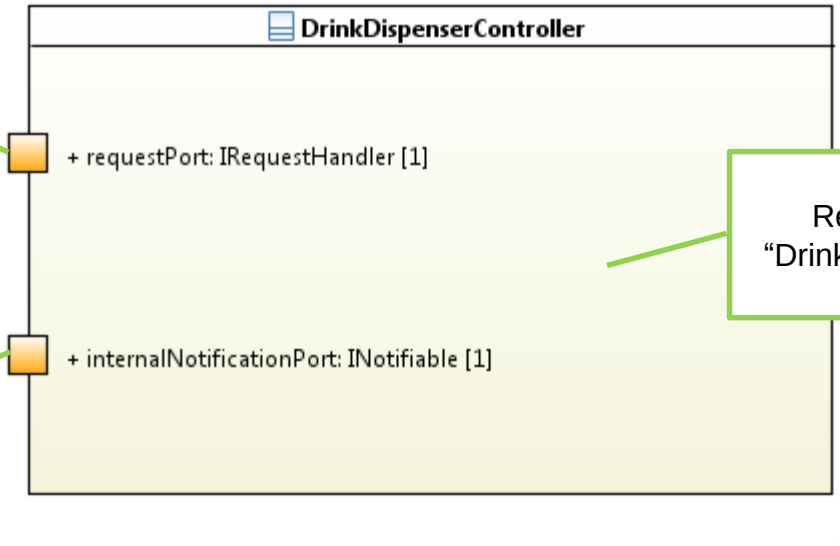
DRINK DISPENSER ARCHITECTURE



BUILD AN EXAMPLE (3/4)

DRINK DISPENSER

Through this port drink requests are received. The interface provided by the port is "**IRequestHandler**"

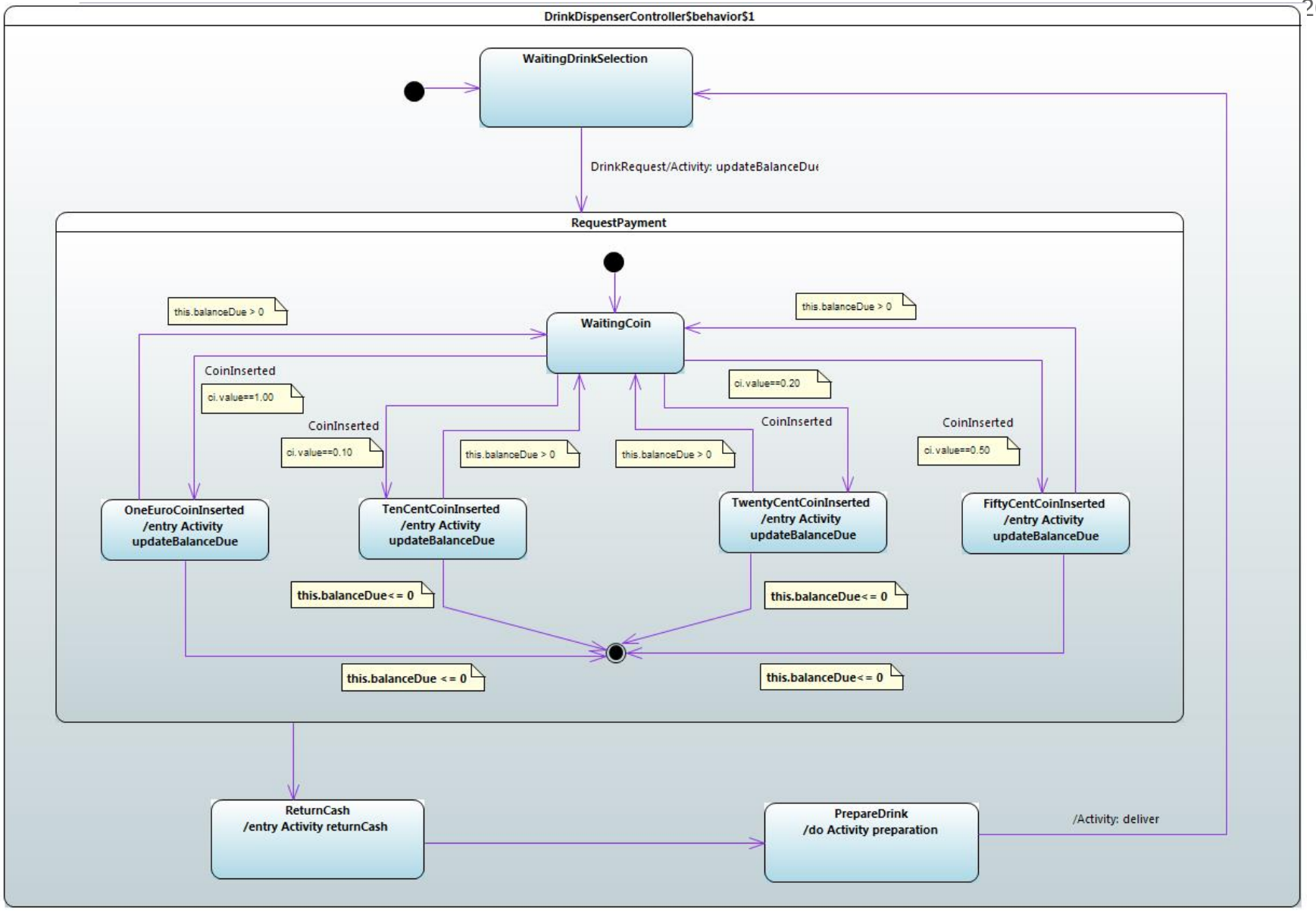


Refinement of class "DrinkDispenserController"

Through this port notifications of the insertion about an inserted coin are received. The interface provided by the port is "**INotifiable**"

BUILD AN EXAMPLE (3/4)

DRINK DISPENSER STATE-MACHINE



Introduction

- Where does UML state-machines come from ?
- Usages of UML state-machines within a model

Active classes and events

- Relation between structure and behaviors

Basic elements to describe behavior state-machines

- State, Transition, IntialPseudoState and FinalState
- Build an example with the subset explained in this section

Advanced behavior state-machines

- Pseudo-states and parallelism
- Refine your example to integrate usage of advanced concepts

Behavior state-machines: understand the global dynamic

- Event dispatching, Run-to-Completion
- Execute step by step your refined example

Going further: advanced semantics of behavior state-machines

- Transition selection, transition conflict, special kind of transition, deferred event, protocol state-machines

Relation between this course and current research

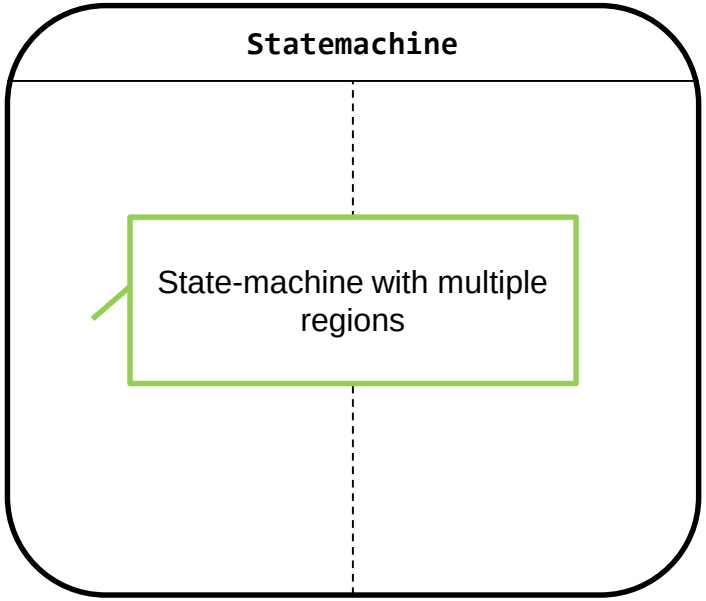
REGIONS (1/3)

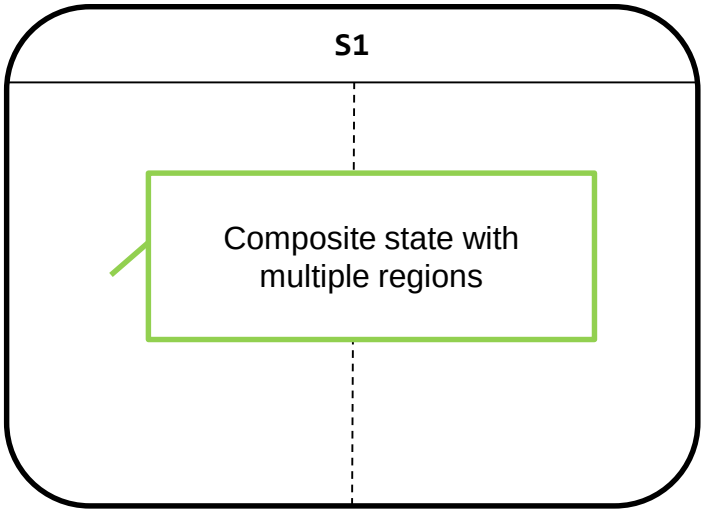
Region

- Used as a top level part of:
  - A state-machine or a composite state
- It acts as a container for vertices and transitions

Region and Parallelism

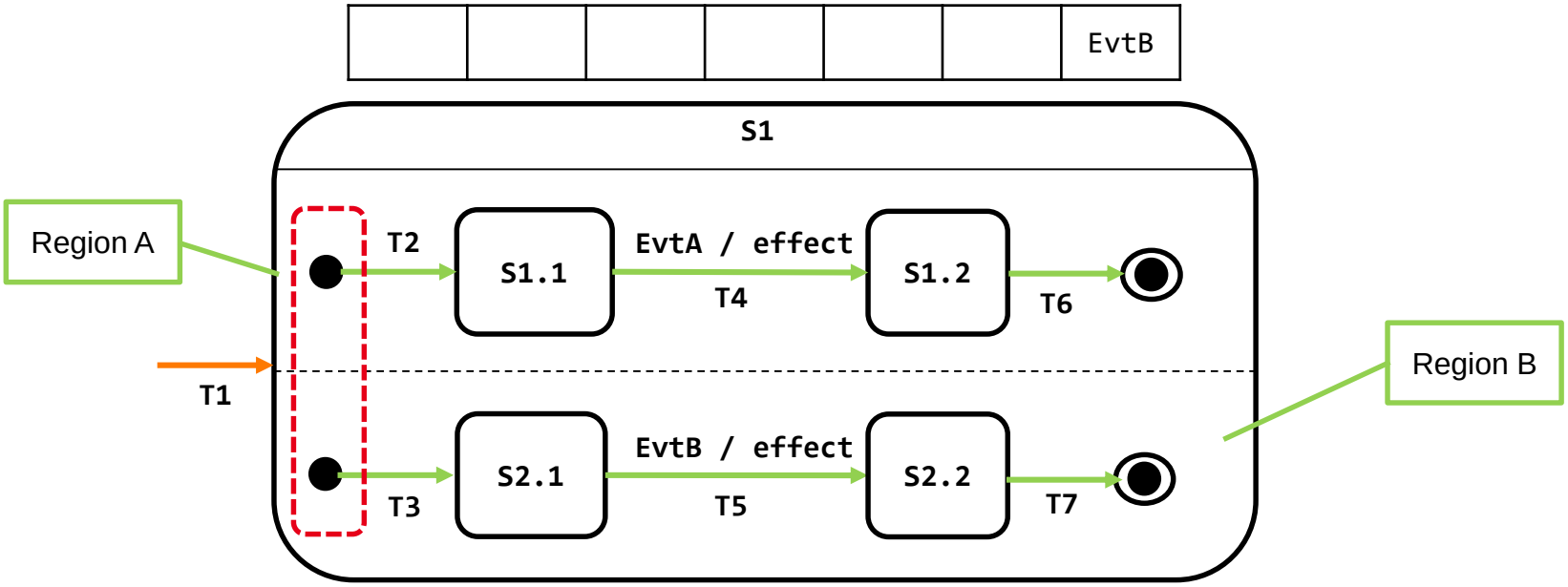
- A state-machine or a composite state can own multiple regions
- Part of the behavior specified inside these regions run in parallel





REGIONS (2/3)

ENTERING A COMPOSITE STATE WITH ORTHOGONAL REGIONS



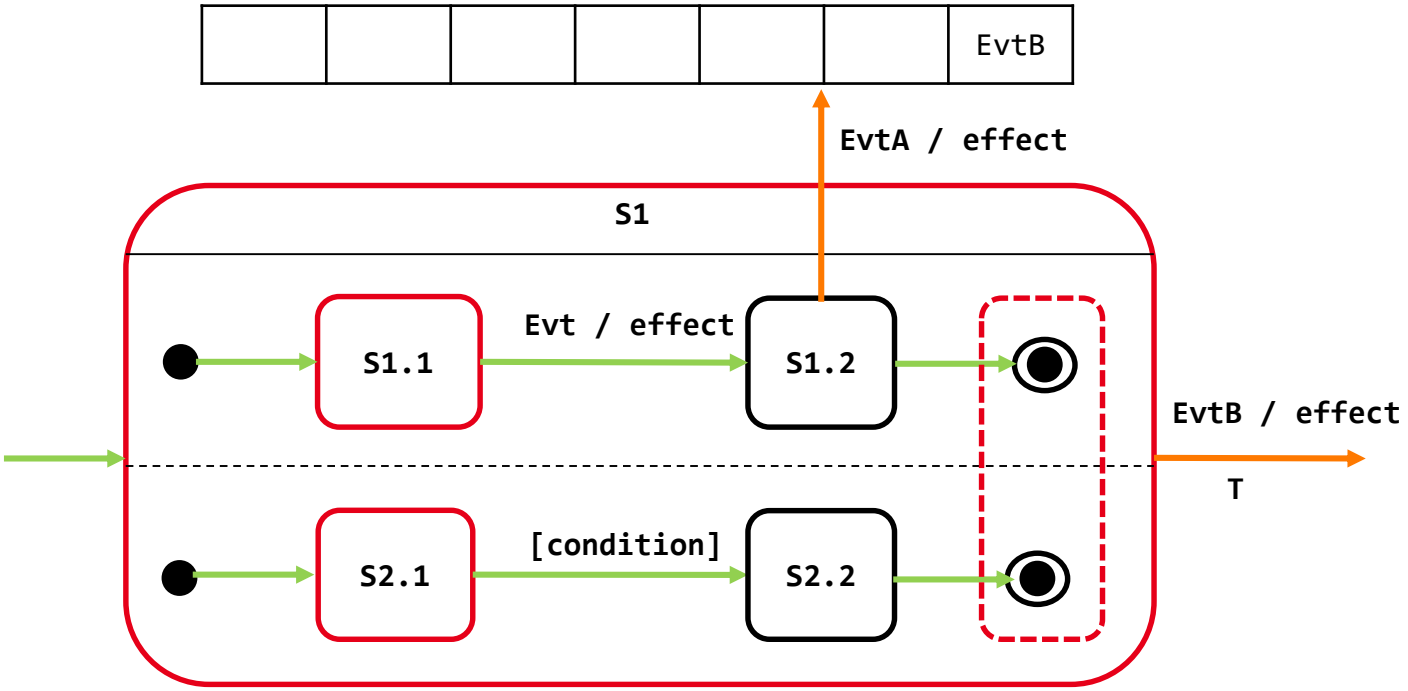
`T1::S1(entered)::[T2::S1.1(entered)||T3::S2.1(entered)]::S2.1(exited)::T5(effect)::…`

Semantics

- When the transition entering S1 fires:
  - Regions A and B are entered
  - Consequently S1.1 and S2.1 are entered in parallel

REGIONS (3/3)

EXITING A COMPOSITE STATE WITH ORTHOGONAL REGIONS



[S1.1(exited)||S2.1(exited)]::S1(exited)::T(effect)::…

Semantics

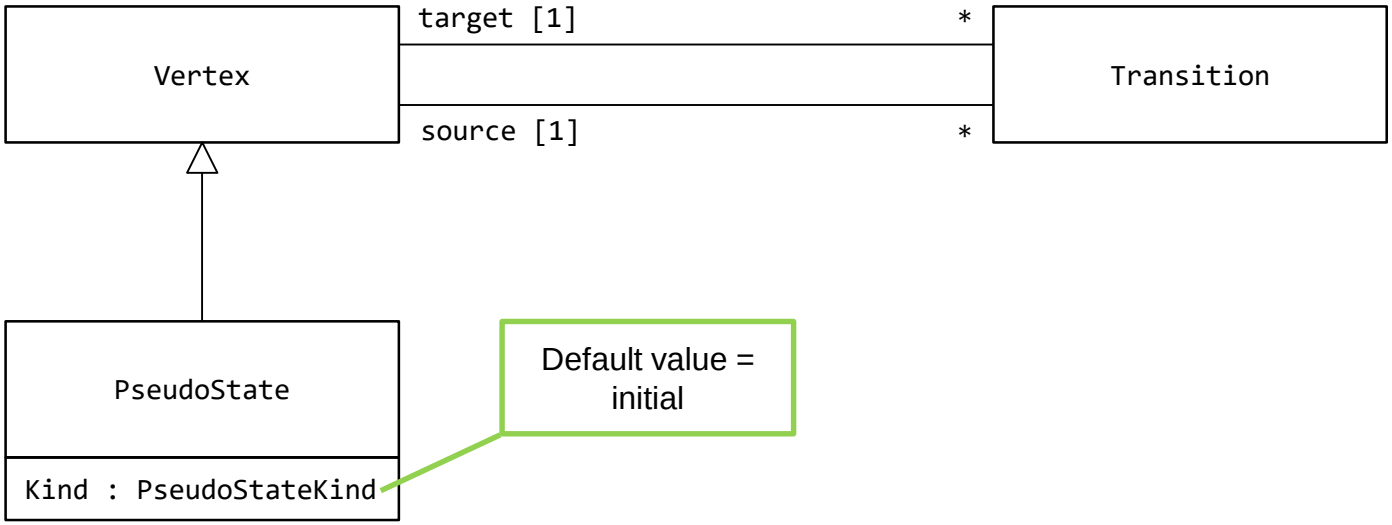
- S1 can be exited when:
  - Both regions reach the final states and one outgoing transition is fireable
  - An outgoing transition of S1 fires

A. Pseudo states role

- Used to chain multiple transitions into more complex « compound transition »
- A pseudo state provides a way to coordinate the execution flow within state machines.

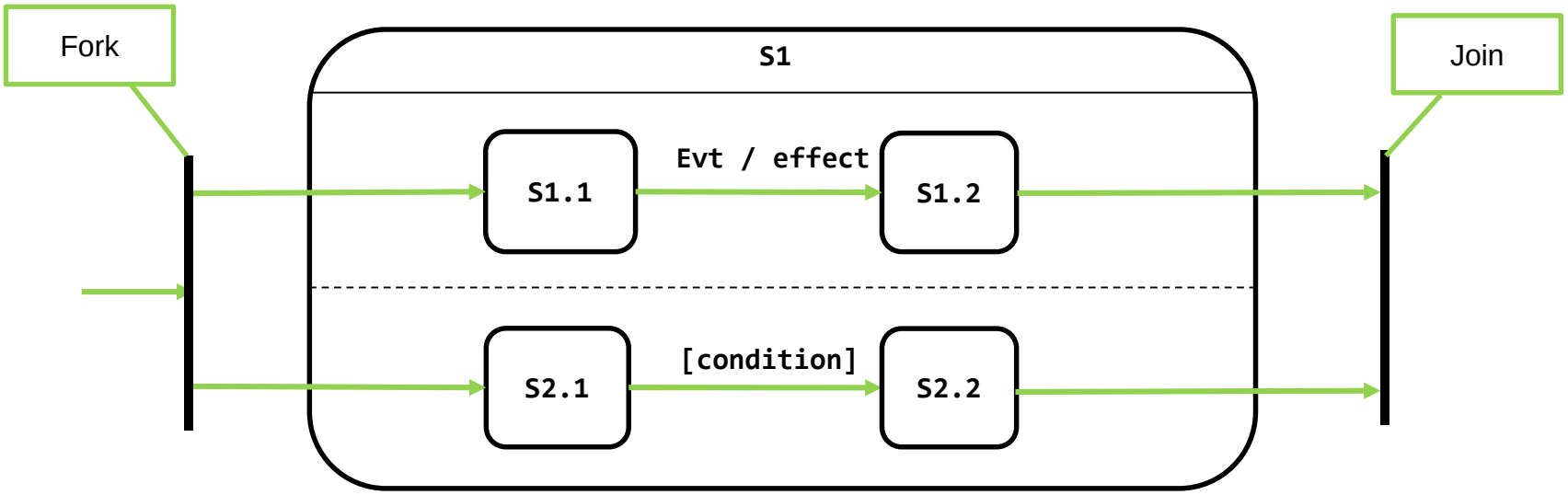
B. What kind of pseudo state does UML provide ?

- Initial, Fork, Join, Choice, Junction, ExitPoint, Entry Point, ShallowHistory, DeepHistory, Terminate



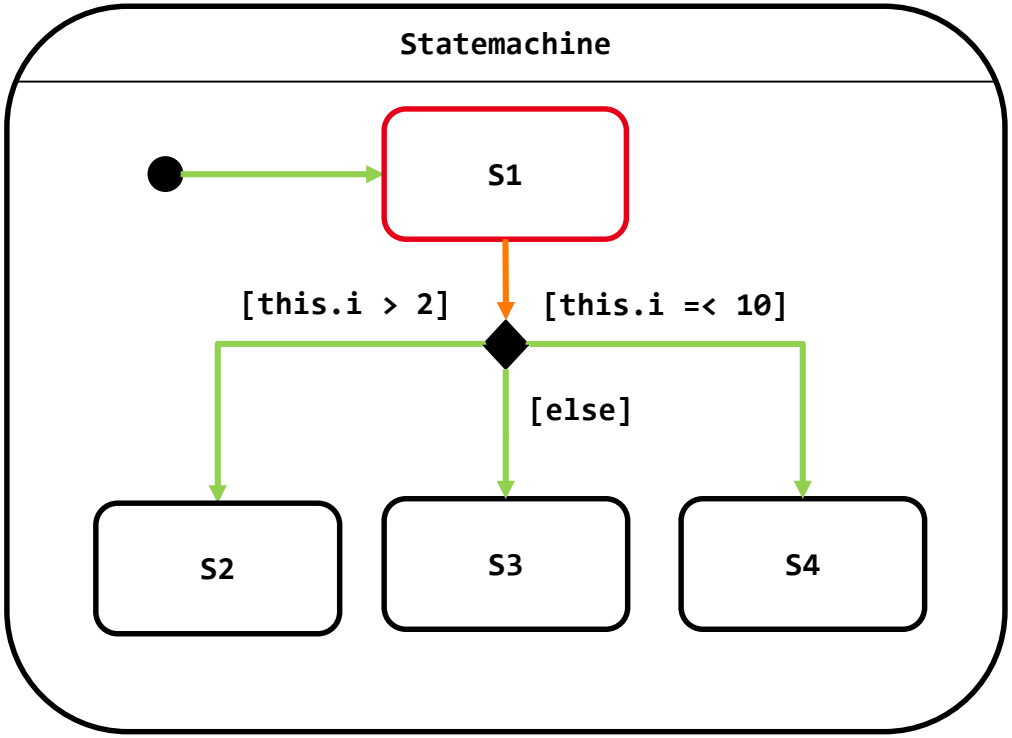
FORK AND JOIN

DESCRIBE PARALLELISM WITH PSEUDO STATES



Semantics

- Fork
  - Split the execution into concurrent execution flows.
  - Both regions get executed
  - S1, S1.1 and S2.1 are entered
- Join
  - Synchronize concurrent execution flows
  - Both regions are exited simultaneously which leads S1 to be exited



Semantics

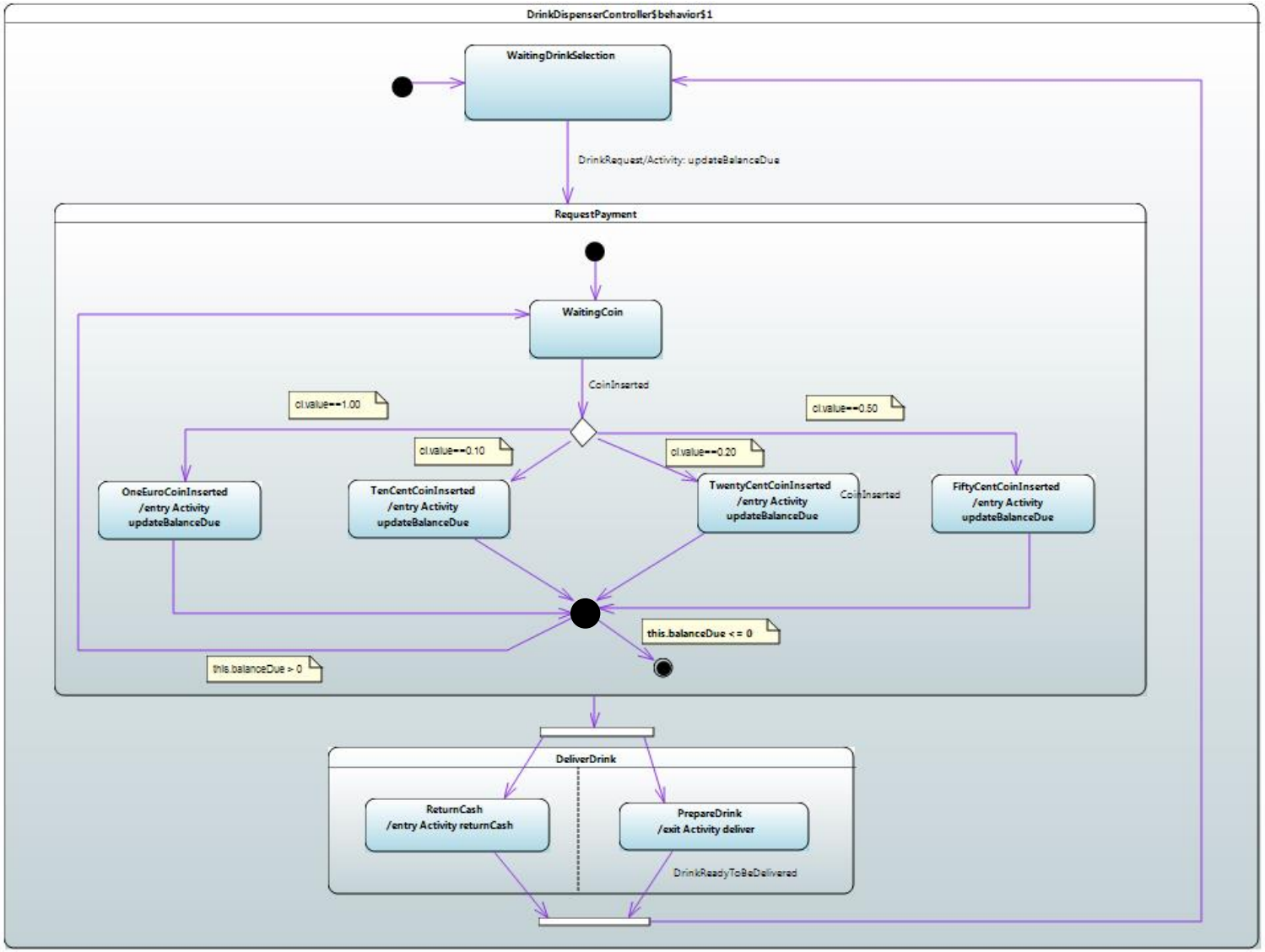
- Choice
  - One path of the alternative is chosen
  - Choice of the path or the other is determined by the evaluation of guards
  - Else guard materialize a default path that is chosen in case no guard evaluate to true.
- Note:
  - If more than a guard evaluates to true then a selection must be realized between fireable transitions. Algorithm for this choice is left undefined

Exercise

- In the state-machine controlling the lifecycle of the DrinkDispenser
  - Introduce the usage of a choice pseudo state
  - Introduce the usage of parallelism

REFINE THE DRINK DISPENSER (2/2)

INTEGRATE USAGE OF PARALLELISM AND PSEUDO-STATES



papyrus

OUTLINE (1/2)

Introduction

- Where does UML state-machines come from ?
- Usages of UML state-machines within a model

Active classes and events

- Relation between structure and behaviors

Basic elements to describe behavior state-machines

- State, Transition, IntialPseudoState and FinalState
- Build an example with the subset explained in this section

Advanced behavior state-machines

- Pseudo-states and parallelism
- Refine your example to integrate usage of advanced concepts

Behavior state-machines: understand the global dynamic

- Event dispatching, Run-to-Completion
- Execute step by step your refined example

Going further: advanced semantics of behavior state-machines

- Transition selection, transition conflict, special kind of transition, deferred event, protocol state-machines

Relation between this course and current research

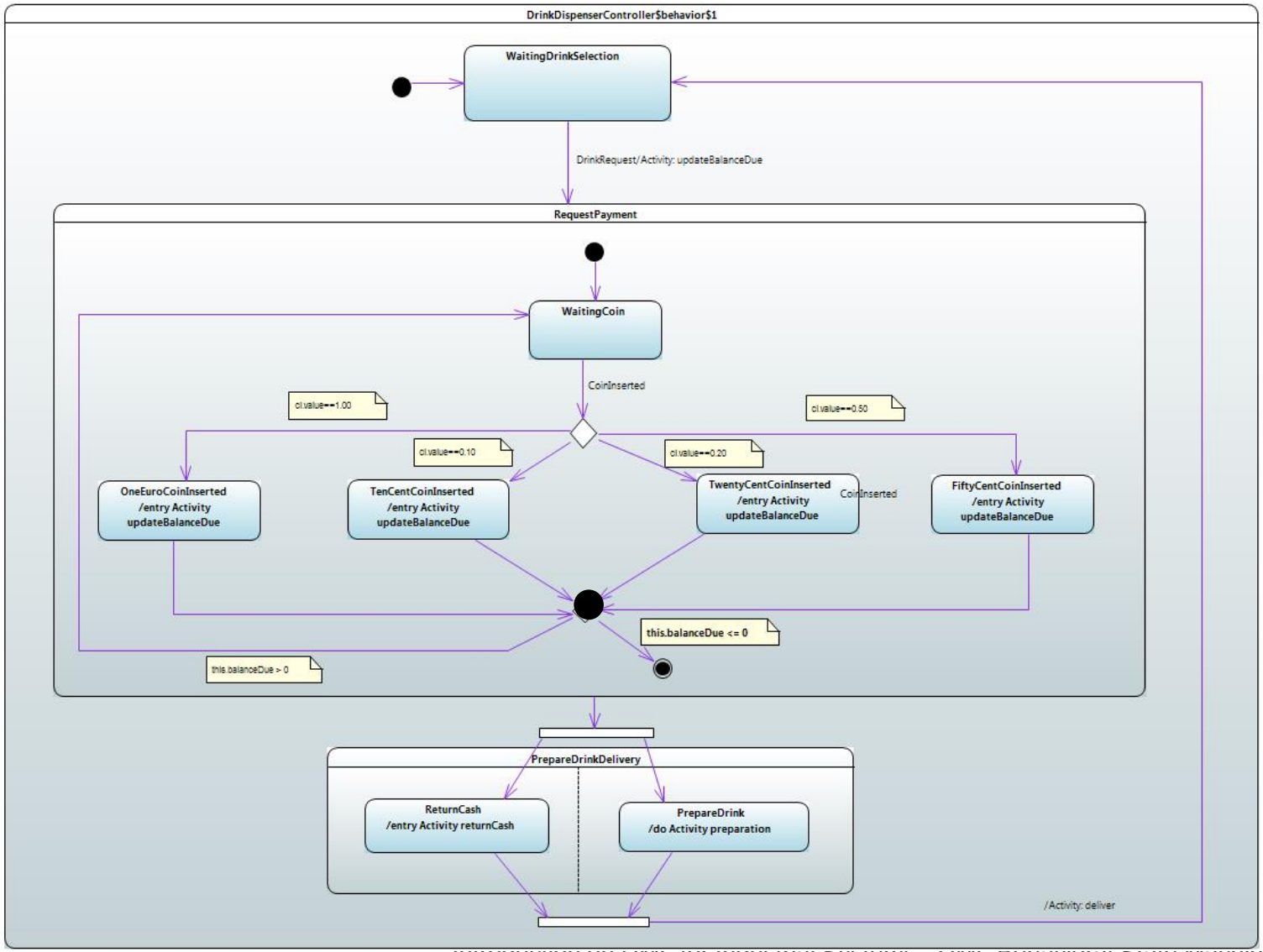
Event pool of an instance of the DrinkDispenser

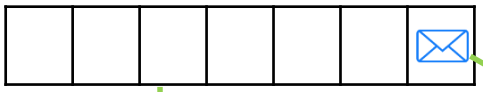
Pepsi

Coffea

Tea

Panel of the DrinkDispenser as it is viewed by the user





Event pool of an instance of the DrinkDispenser

Pepsi

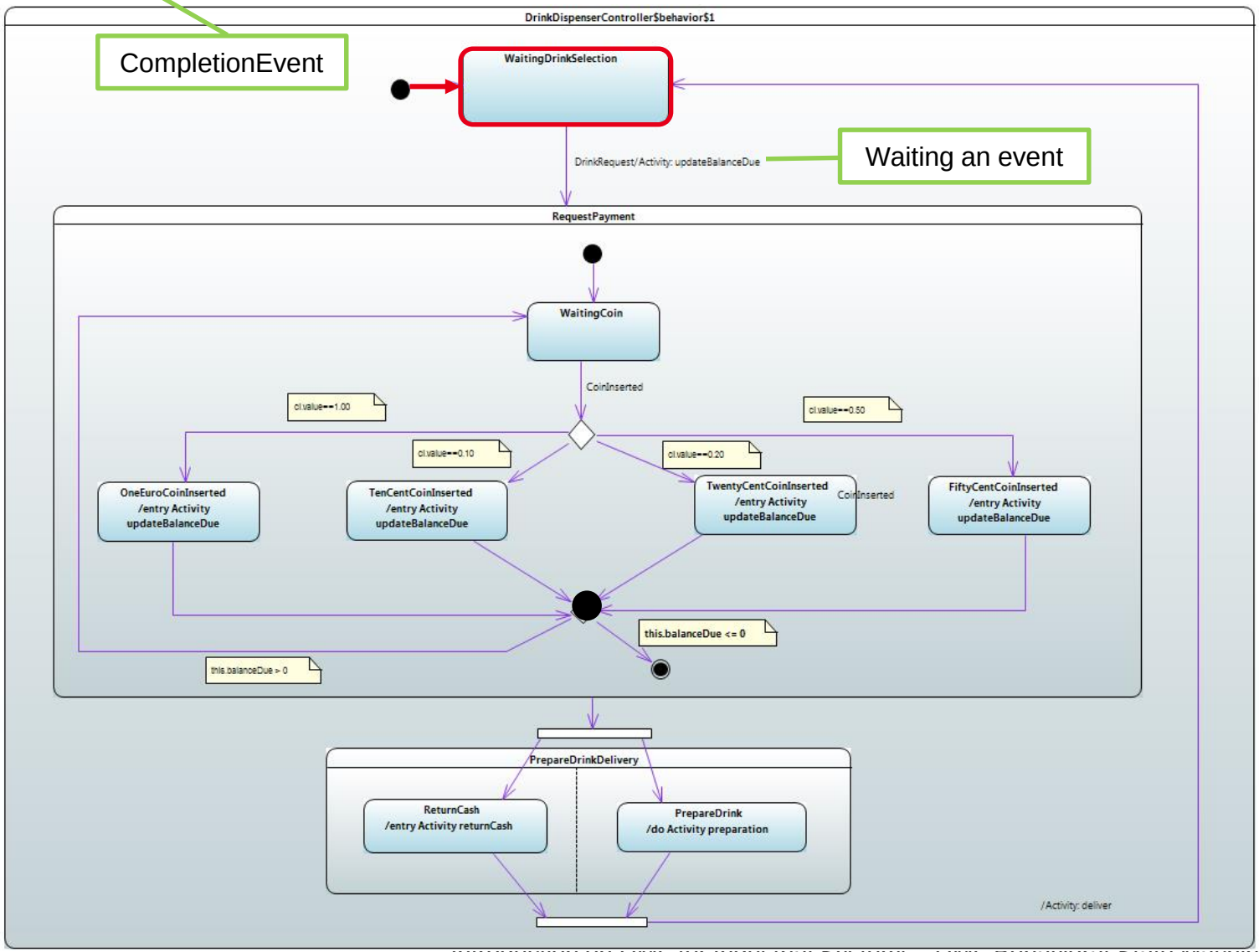
Coffea

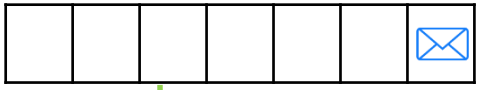
Tea

Panel of the DrinkDispenser as it is viewed by the user

CompletionEvent

Waiting an event





No completion transition available. The completion event is lost

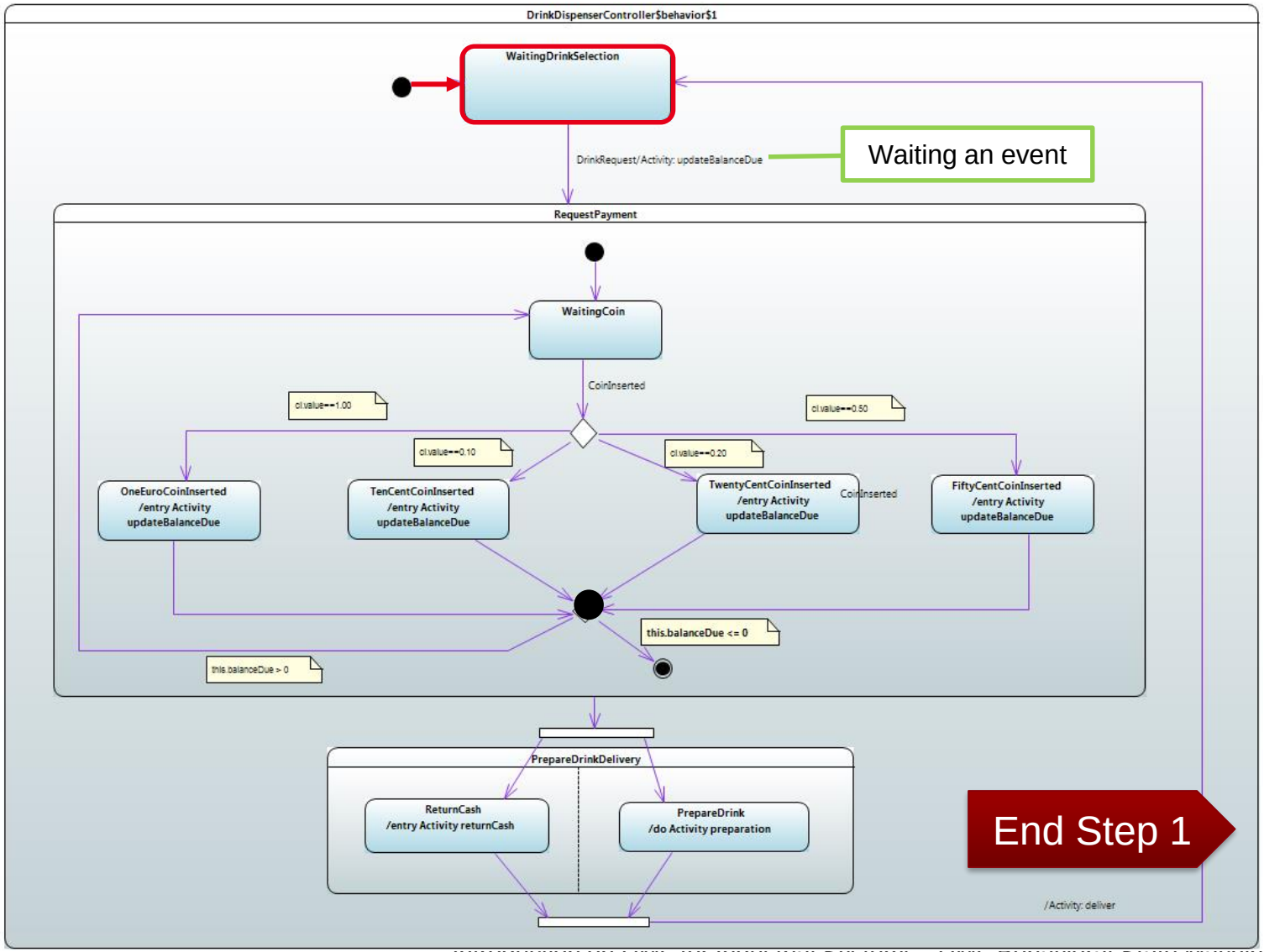
Event pool of an instance of the DrinkDispenser

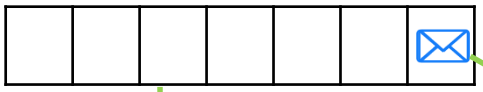
Pepsi

Coffea

Tea

Panel of the DrinkDispenser as it is viewed by the user





The consumption of the signal triggers a set of reactions

CompletionEvent

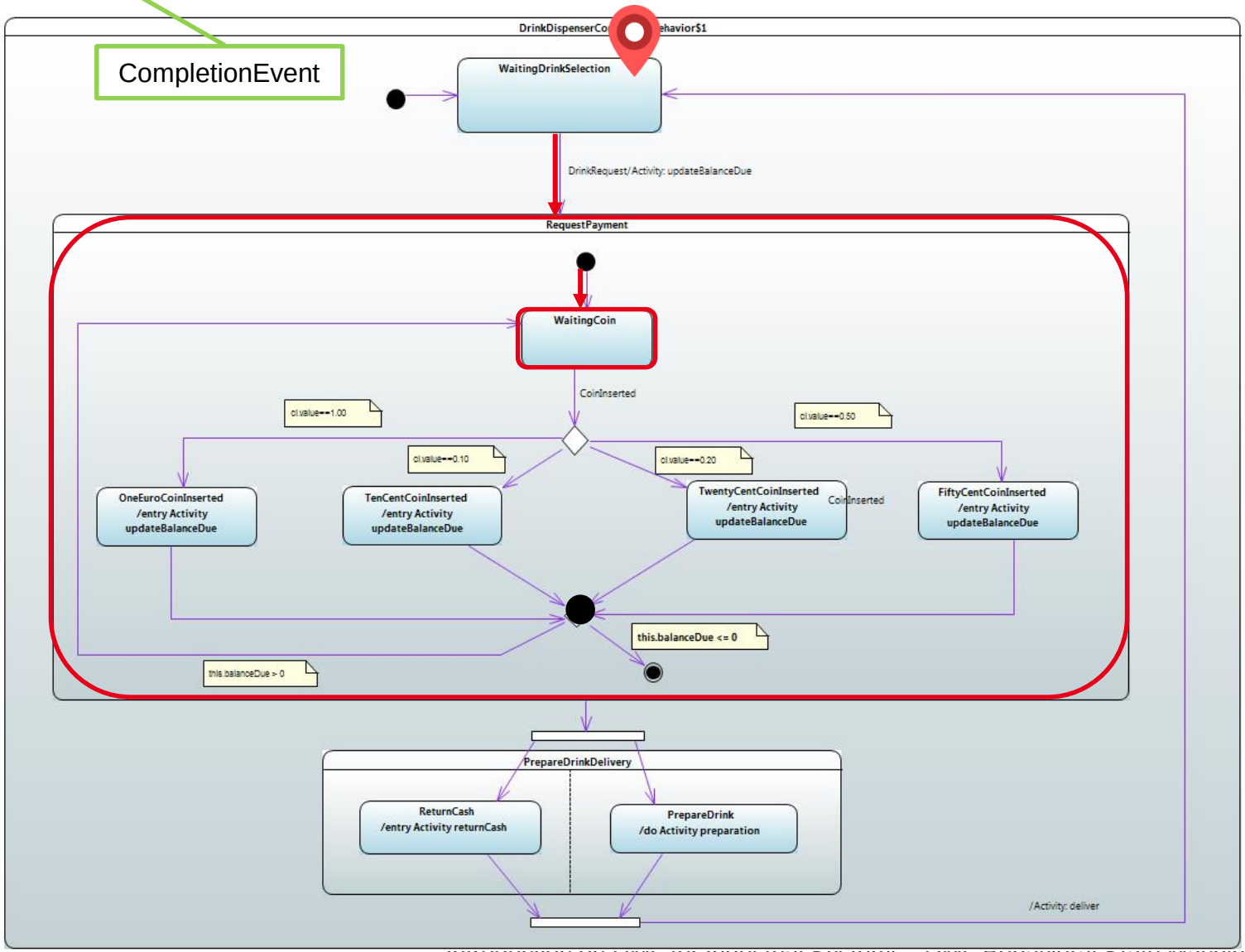
Event pool of an instance of the DrinkDispenser

Pepsi

Coffea

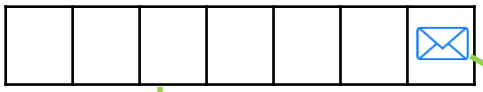
Tea

Panel of the DrinkDispenser as it is viewed by the user



STATE-MACHINE DYNAMICS

STEP 2

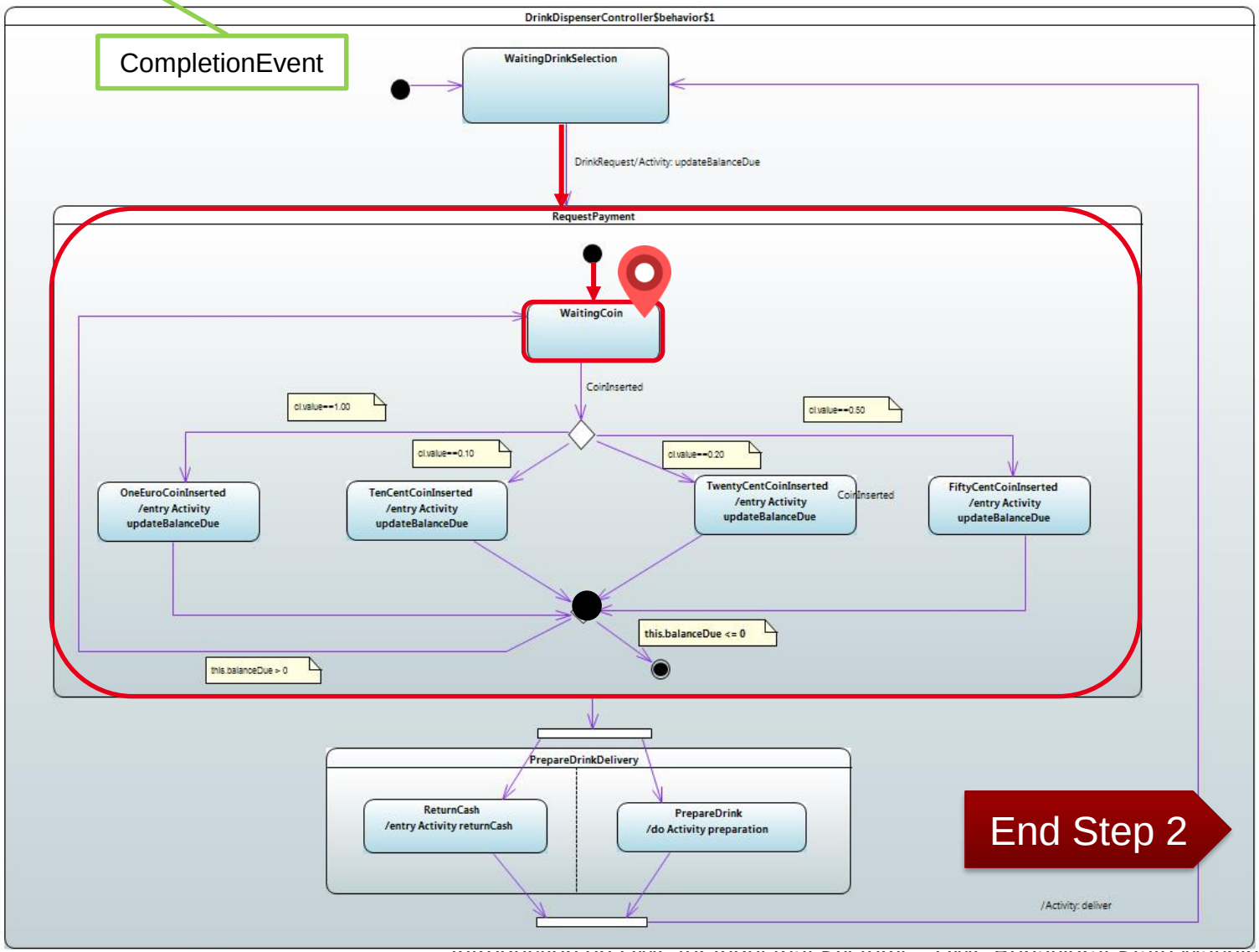


No completion transition available. The completion event is lost

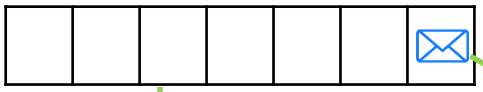
CompletionEvent

Event pool of an instance of the DrinkDispenser

Panel of the DrinkDispenser as it is viewed by the user



End Step 2



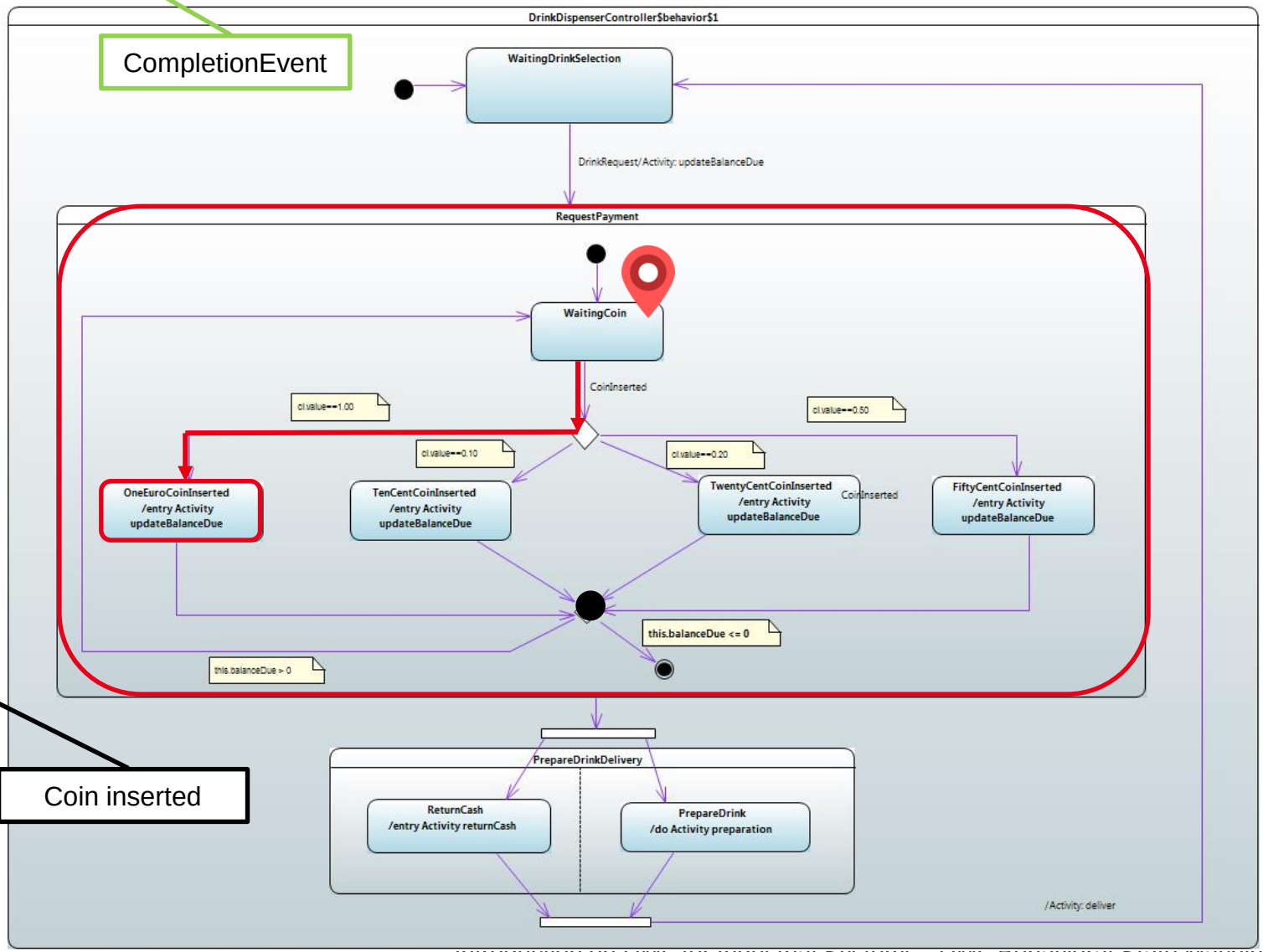
The consumption of the signal triggers a set of reactions

CompletionEvent

Event pool of an instance of the DrinkDispenser

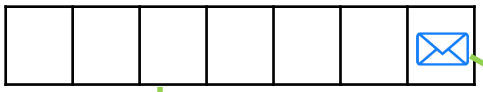
Coin inserted

Panel of the DrinkDispenser as it is viewed by the user



STATE-MACHINE DYNAMICS

STEP 3

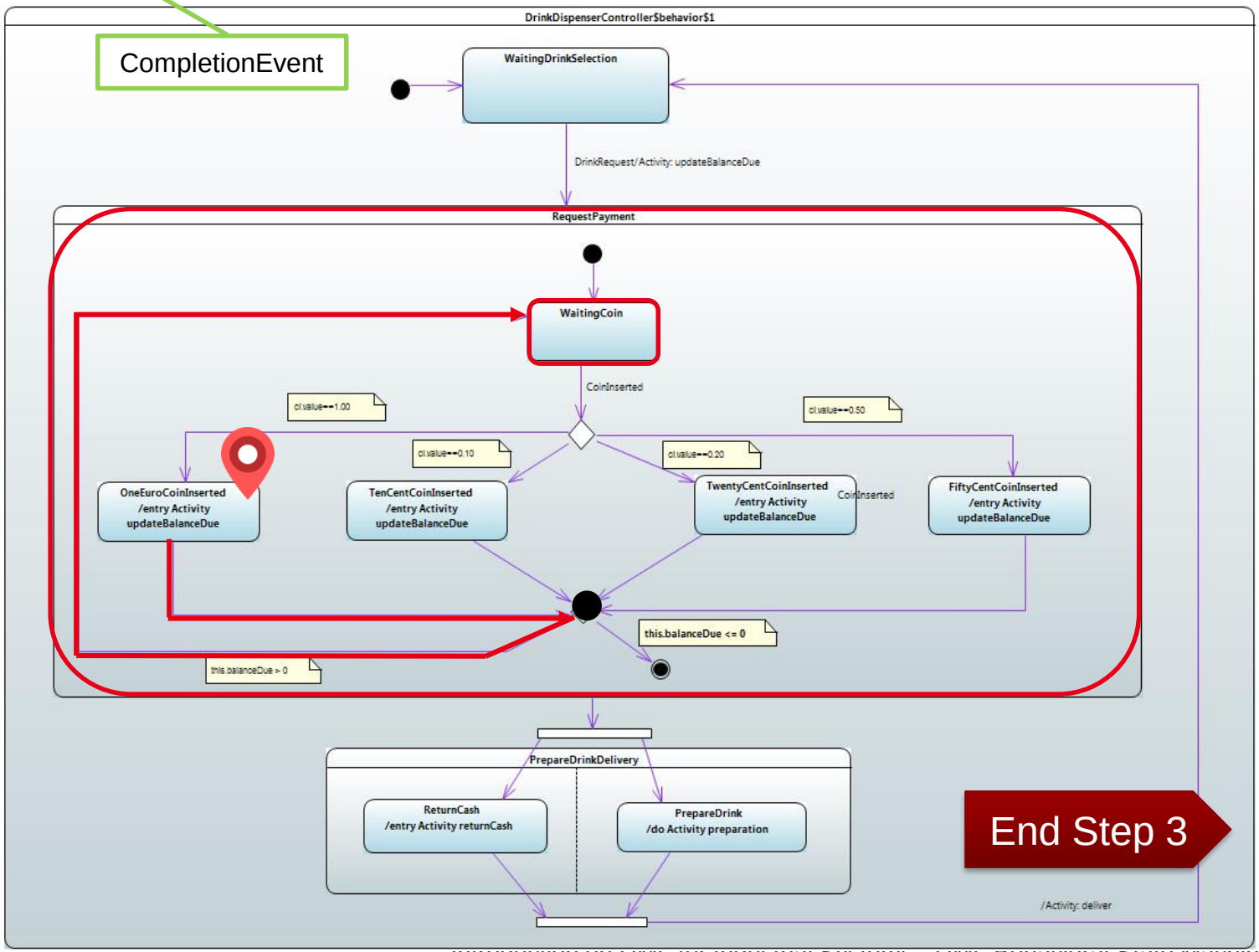


One transition can fire on the arrival of a completion event

CompletionEvent

Event pool of an instance of the DrinkDispenser

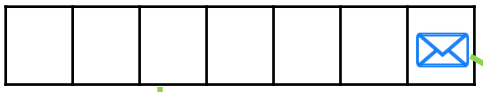
Panel of the DrinkDispenser as it is viewed by the user



End Step 3

A. Step 4
- A coin of 20ct is entered
- 10 cent are still missing to get the soda
- Step ends on the « WaitingCoin » state

B. Step 5
- A coin of 20 cent is entered
  - Imply we need to provide cash back to the user
- A completion event is generated when entering final state of « RequestPayment »
- The next slide describes the consumption of the completion event

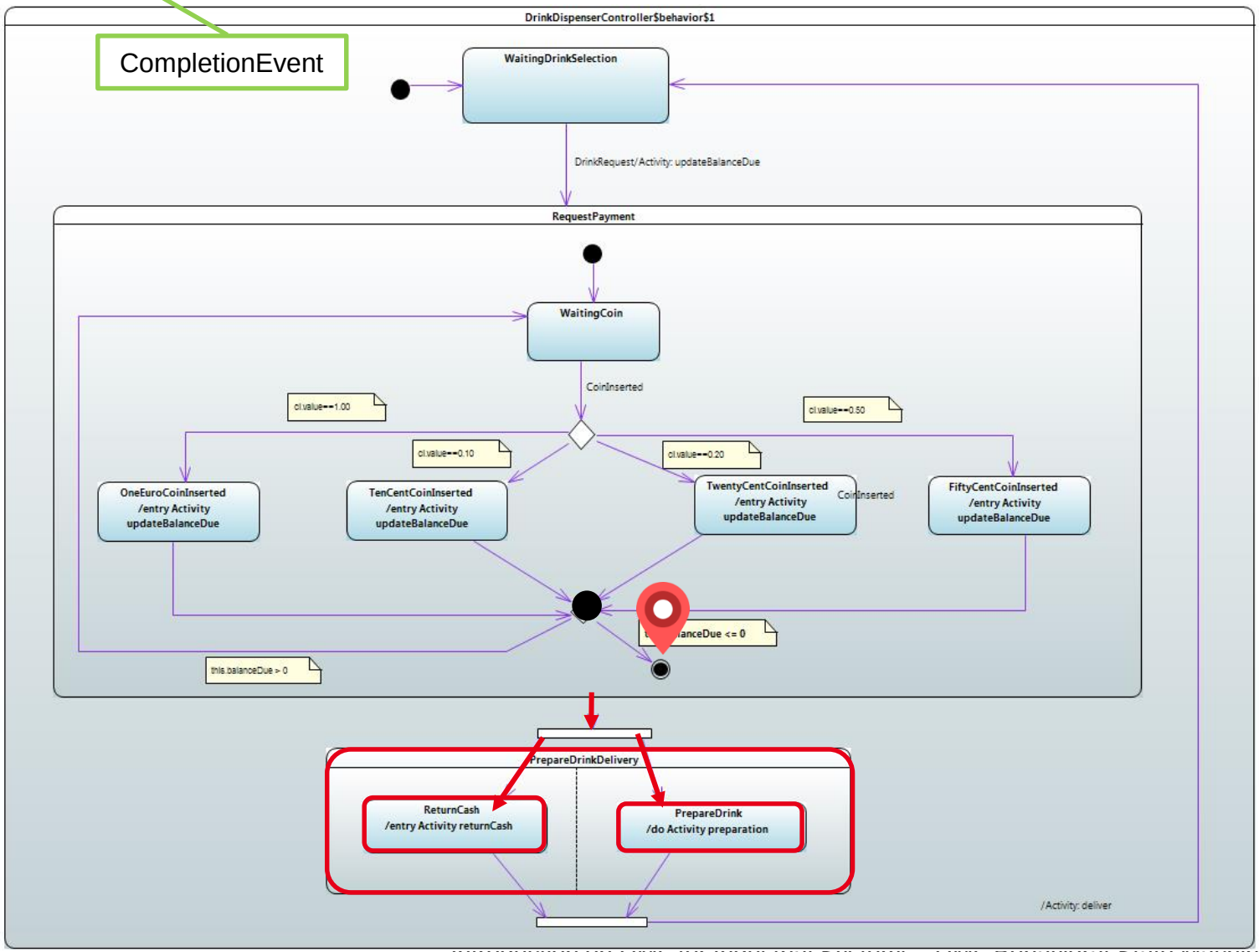


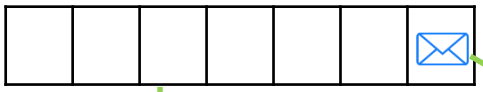
One transition can fire on the arrival of a completion event

CompletionEvent

Event pool of an instance of the DrinkDispenser

Panel of the DrinkDispenser as it is viewed by the user





One transition can fire on the arrival of a completion event

CompletionEvent

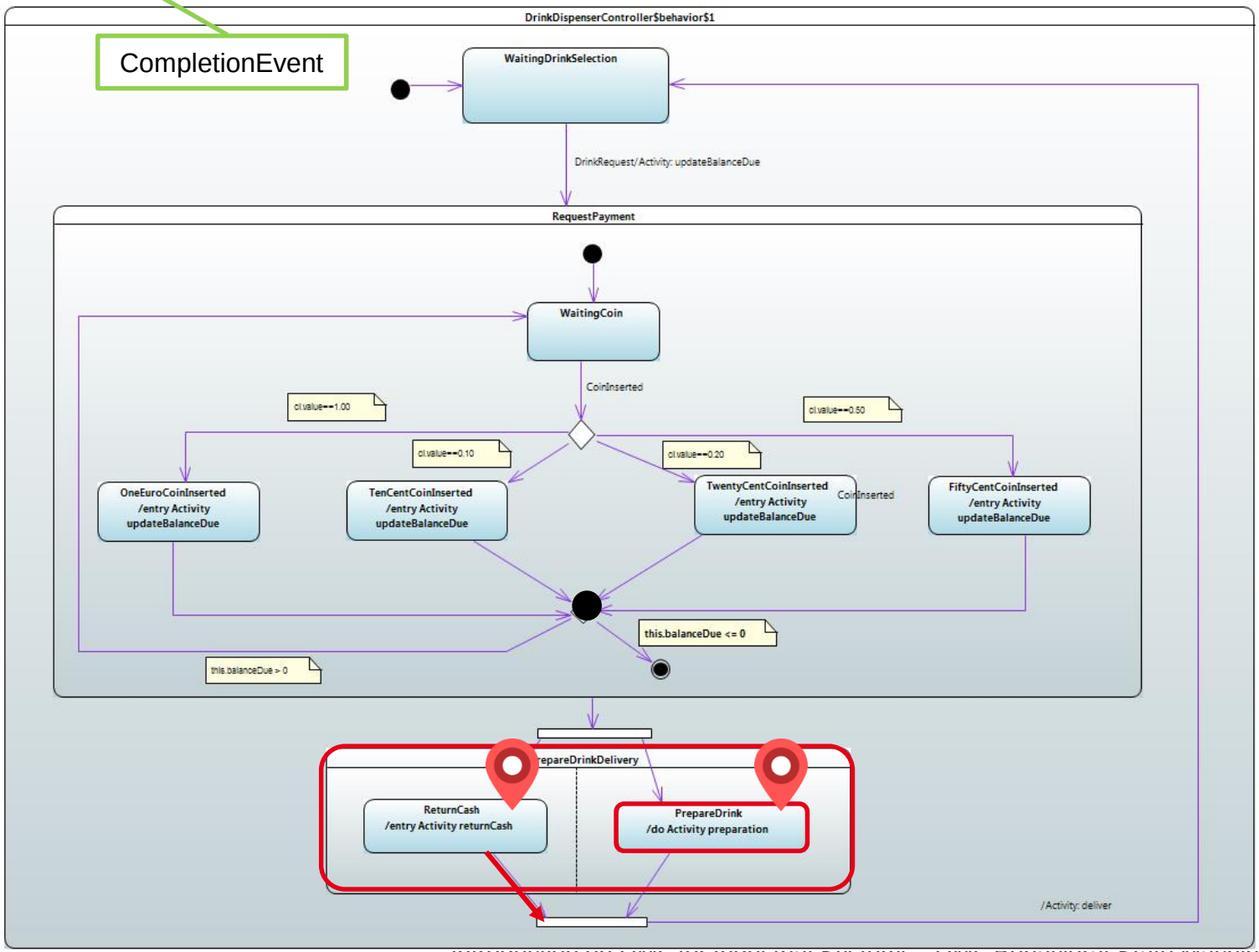
Event pool of an instance of the DrinkDispenser

Pepsi

Coffea

Tea

Panel of the DrinkDispenser as it is viewed by the user



A. Event dispatching

- Take out an event from the event pool.
  - One at a time
  - The policy of event consumption is FIFO (First In First Out)
  - CompletionEvent and others: SignalEvent, CallEvent, ChangeEvent, etc.
- Find a transition having a trigger that matches the event
  - If no transition match then the event is lost
  - If more than a transition match it may exist a conflict
  - If a single transition matches this latter is selected

B. Run to completion

- Starts with the event dispatching process
- It continues while the state-machine did not reach a stable state
  - A stable state is the situation where not any transition can fire
- The end of the RTC step leads to
  - Dispatch another event if available (next step)
  - Or to suspend the context classifier until the arrival of another event

Introduction

- Where does UML state-machines come from ?
- Usages of UML state-machines within a model

Active classes and events

- Relation between structure and behaviors

Basic elements to describe behavior state-machines

- State, Transition, IntialPseudoState and FinalState
- Build an example with the subset explained in this section

Advanced behavior state-machines

- Pseudo-states and parallelism
- Refine your example to integrate usage of advanced concepts

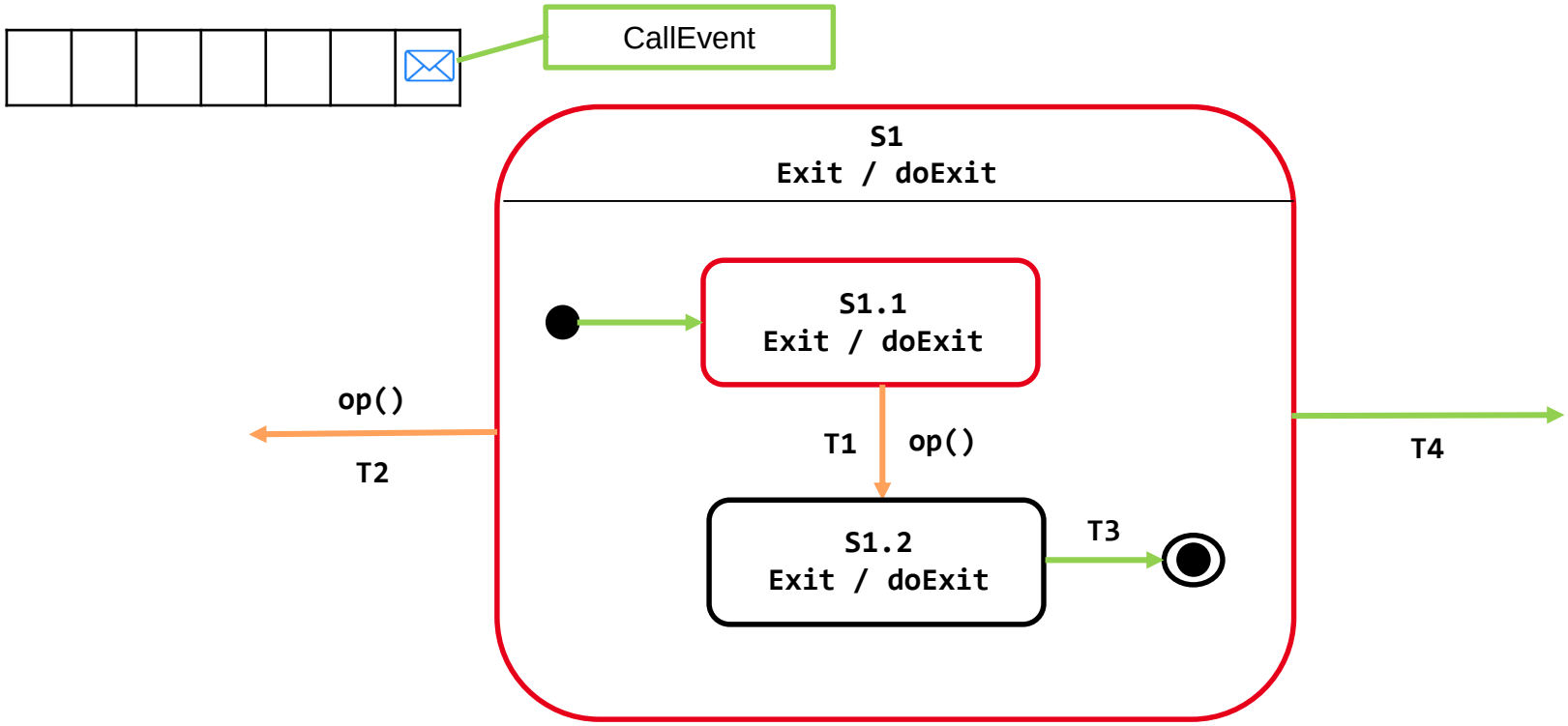
Behavior state-machines: understand the global dynamic

- Event dispatching, Run-to-Completion
- Execute step by step your refined example

Going further: advanced semantics of behavior state-machines

- Transition selection, transition conflict

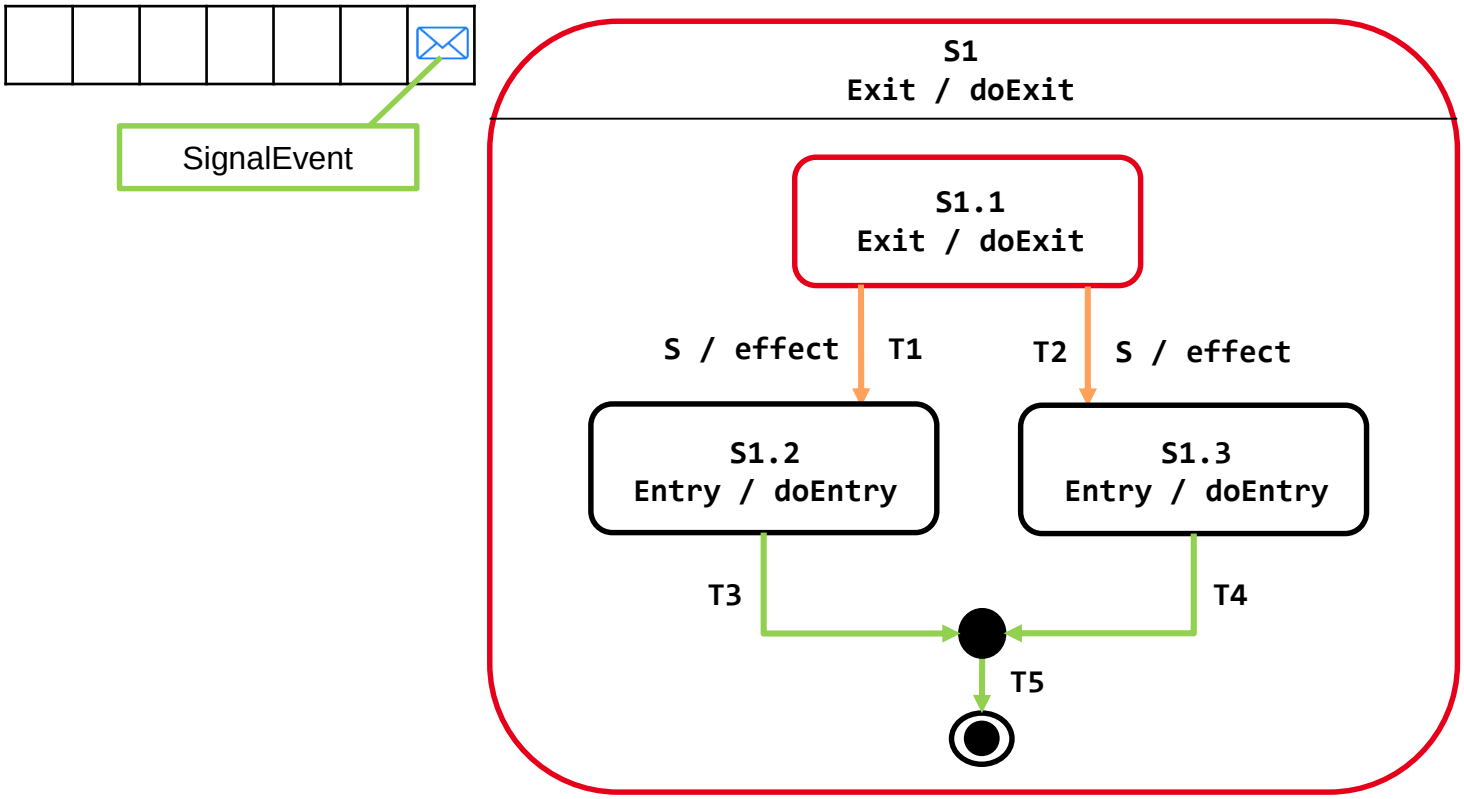
Relation between this course and current research



Hierarchical priorities

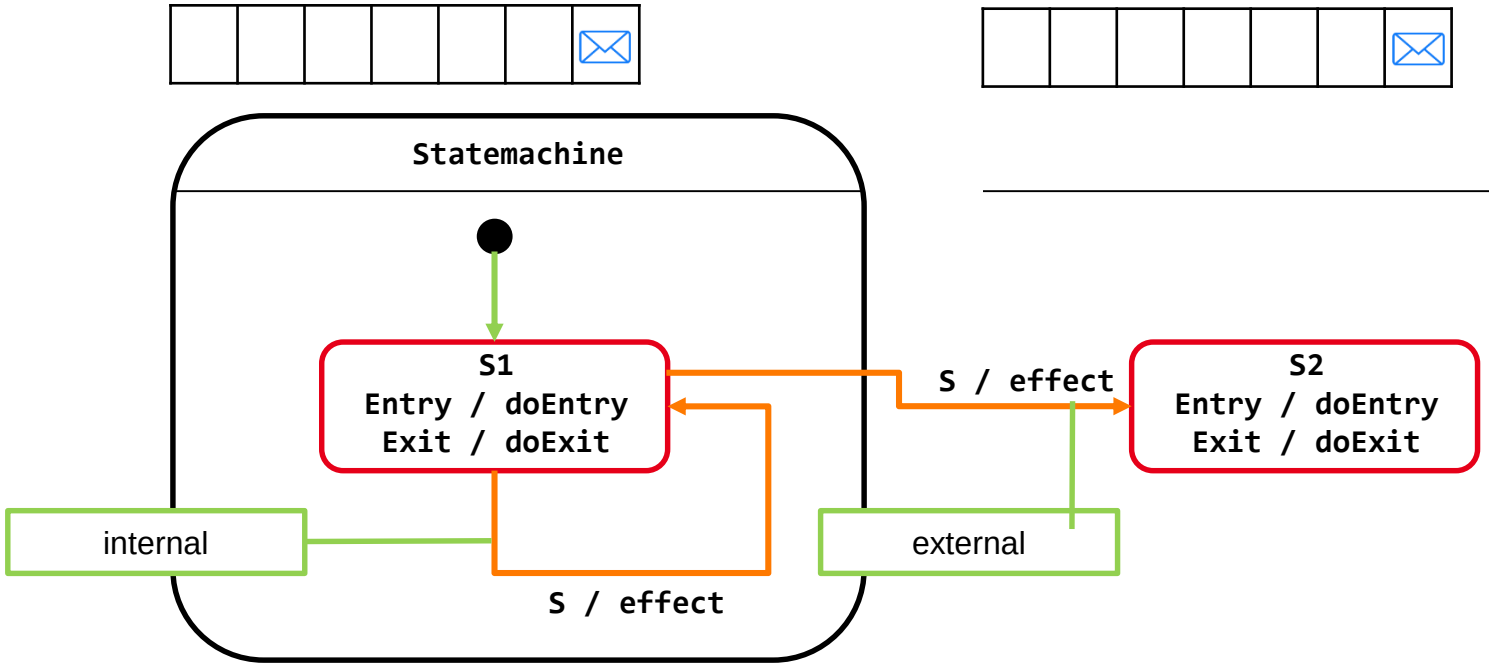
- Which transition will be executed ? T1 or T2 ?
  - There is no ambiguity here
  - T1 has the priority over T2
  - The evaluation of fireable transitions starts with the innermost active state

```
S1.1(doExit)::T1(op)::S1.2(doExit)::T3::FinalState::S1(doExit)::T4
```



Conflicting transitions

- Which transition will be executed ? T1 or T2 ?
  - Both can react to S and so both can fire exiting the same state
  - Hierarchy rules do not help to find a resolution to the conflict
  - UML 2.5 does not provide answer. This problem has to be solved at the semantic level by the tool vendors with the risk that they implement different resolutions (e.g. choose randomly one transition, arbitrarily choose first one).



Playing with transition kind

- External (when the transition fires)
  - The exit behavior of the state is executed
  - The effect behavior is executed
  - The entry behavior of the state is executed
- Internal (when the transition fires)
  - Only the effect of the transition is executed


papyrus
www.eclipse.org/papyrus


Moka