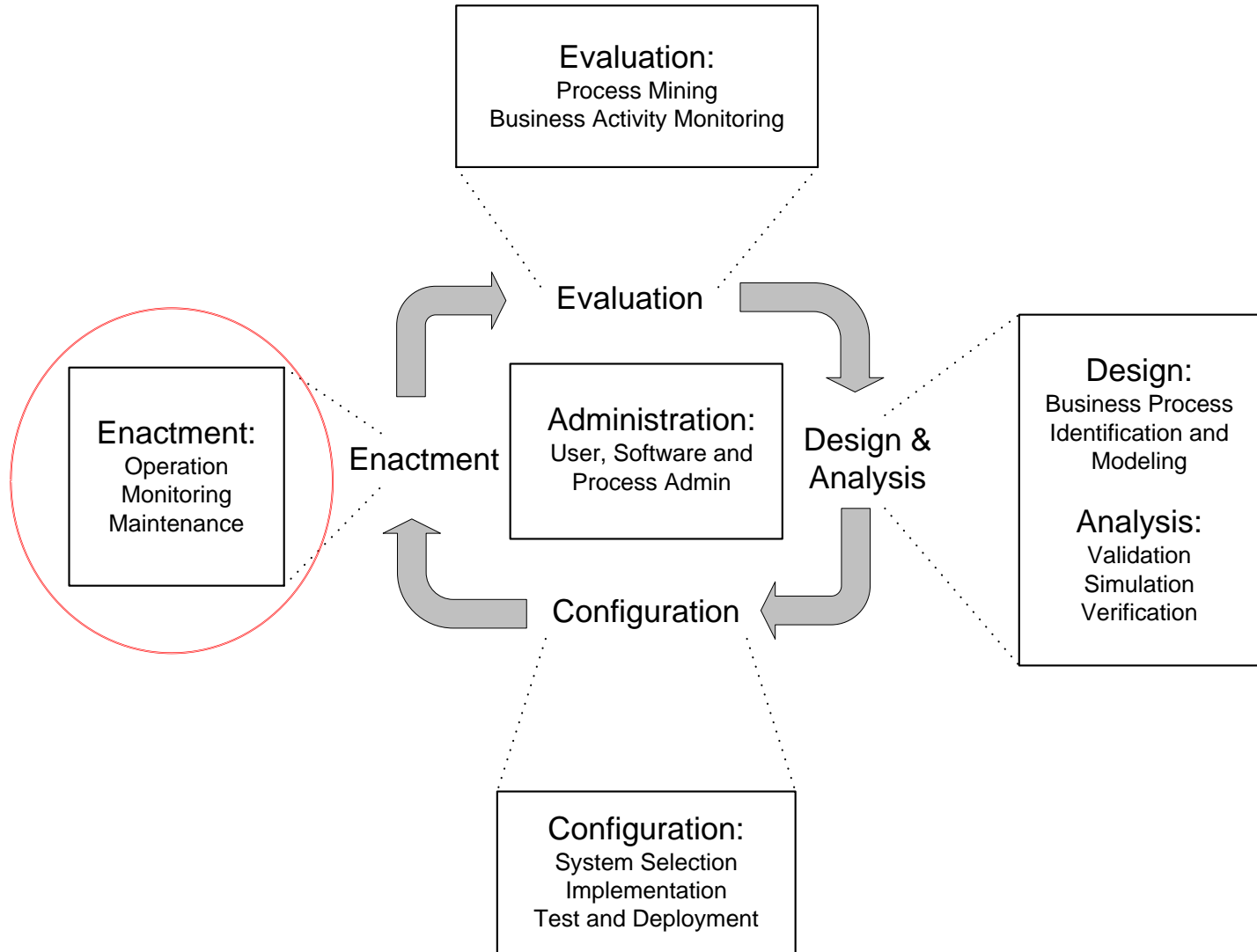


Topic 3: Process Instantiation



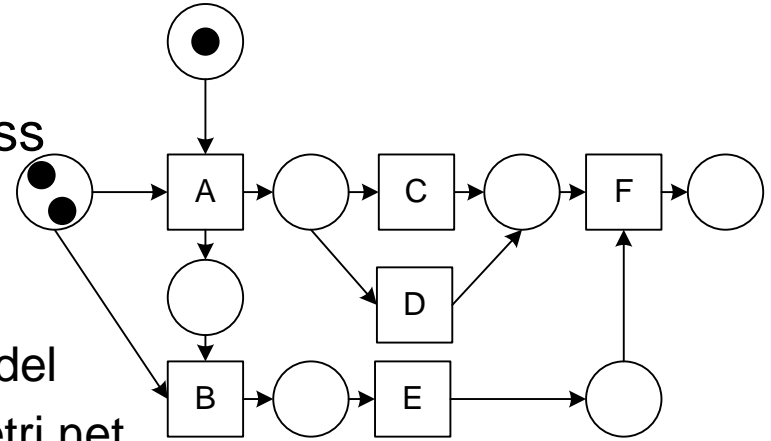
from M. Weske: Business Process Management, © Springer-Verlag Berlin Heidelberg 2007

Instantiation of Processes

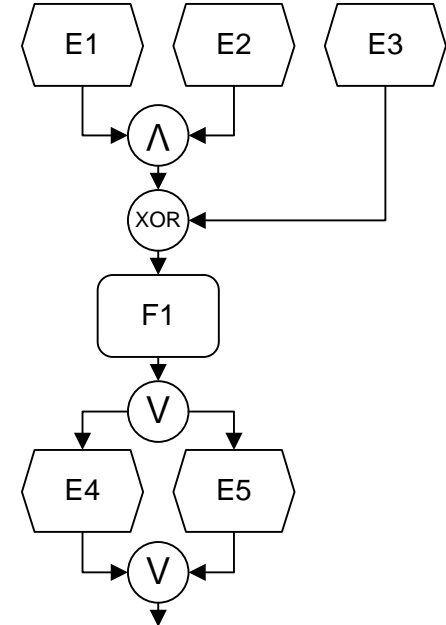
- Process instantiation
 - Central question: When a process instance is created?
 - What rules are to be considered?
- Clear Instantiation semantics is crucial
 - To allow a clear interpretation of the model
 - Specifically: To know when a process should be started
- Hint
 - The adoption of structural soundness is abandoned here

Start State

- Start state is initial state of each process instance
- Explicit start state
 - Part of the definition of the process model
 - For example, the initial marking of a Petri net
 - Satisfied for structural soundness



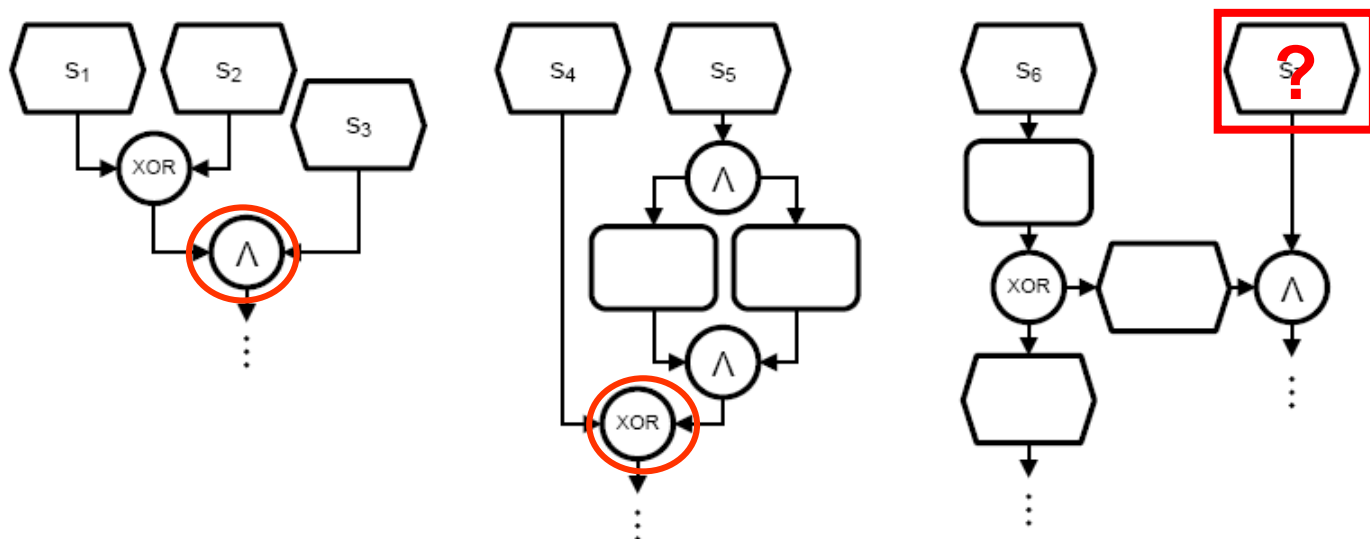
- Implicit start state
 - Is derived from the structure of the process model
 - Process model has entry points
 - These characterize the initial state (s)
 - For example, events with no incoming edges in an EPC



Deriving the initial state

- Start-Join
 - Join nodes, so all paths from entry points to endpoints contain that node

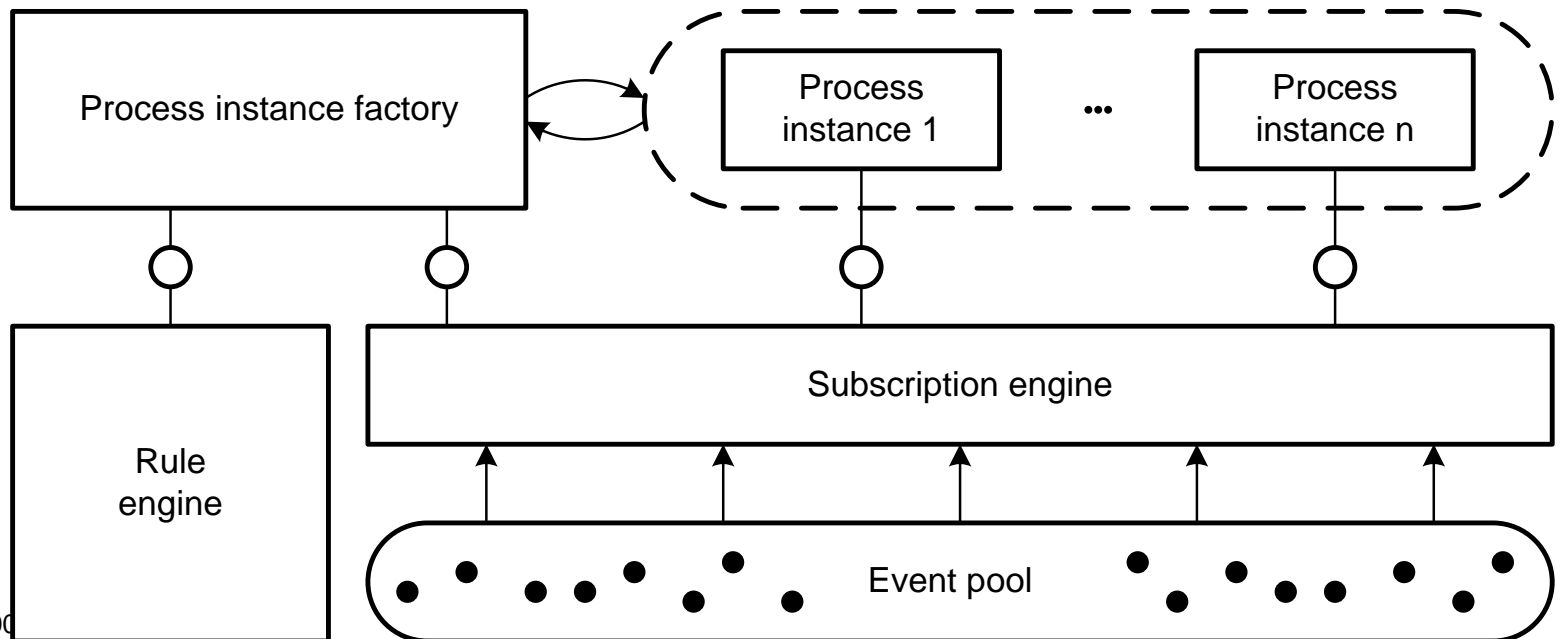
- Observation
 - If there are several start-joins, we pick the first
 - I.e., there is path to each other start-join
 - Based on the minimal start-join, valid starting states can be identified



Technical View on Start Events

- If instantiation includes start events
 - Acceptance of an event-processing infrastructure
 - Instance Creation by Process Instance Factory

- Subscription
 - *Process Instance Factory subscribes for start events*
 - Process instances subscribe for process-related events



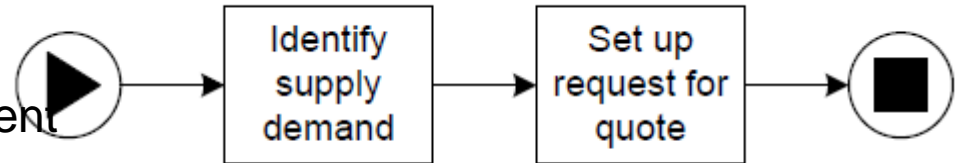
CASU Framework

- CASU Framework answers the questions
 - What are the possibilities of instantiation?
 - How those are supported by process modeling languages?
- **Creation (C):**
When is an instance created?
- **Activation (A):**
Which entry points are activated immediately after instantiation?
- **Subscription (S):**
For which non-activated start events are subscriptions created?
- **Un-subscription (U):**
How long are subscriptions kept?

CASU: Creation

- **C-1 Ignorance**

- Not start condition, no start event
- The process environment controls instantiation
- In the example, it is not clear under which conditions the supply demand is identified



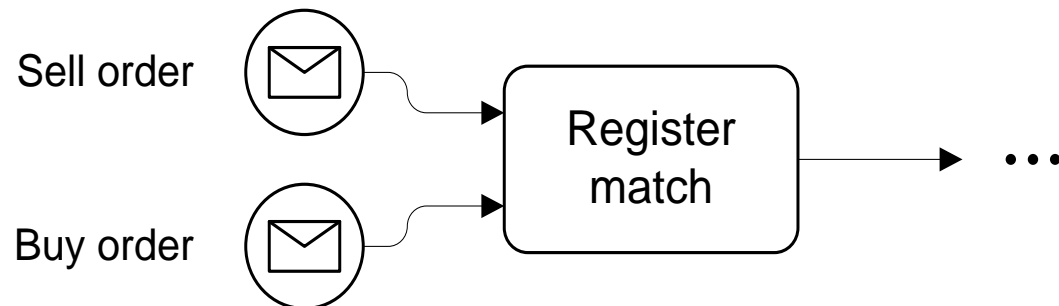
- C-2 Single Event Trigger

- C-3 Multi Event Trigger



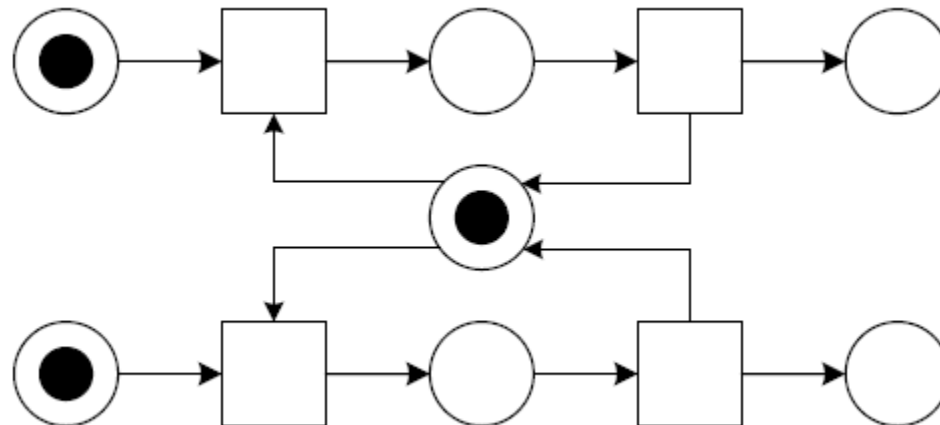
CASU: Creation

- C-1 Ignorance
- **C-2 Single Event Trigger**
 - A Police process model describes that citizens can file charges via a website, triggering instantiation by submitting the web form.
- **C-3 Multi Event Trigger**
 - Creation by means of events (analog C-4), which are consumed at the start
 - Multiple start events are required
 - If all of the necessary events are present, instantiation succeeds



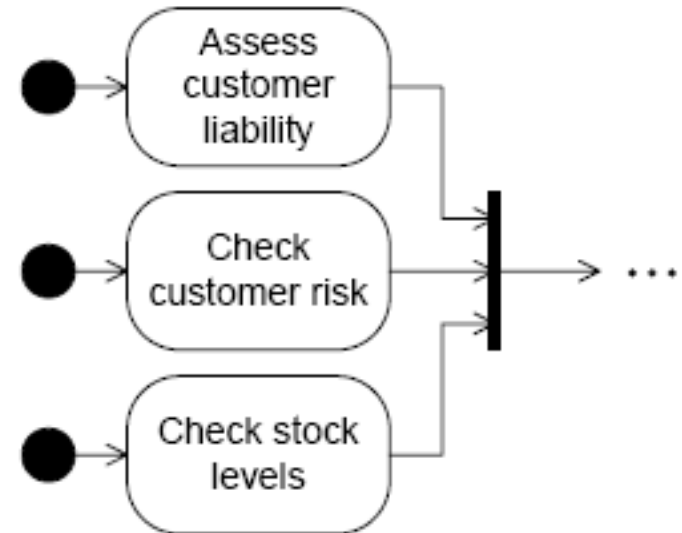
CASU: Activation

- **A-1 Initial State**
 - Explicit definition of the initial state of a process instance
 - For example, a Petri net system
- A-2 All Start Places
- A-3 True Conditions
- A-4 Occurred Events
- A-5 Occurred Events and Conditions



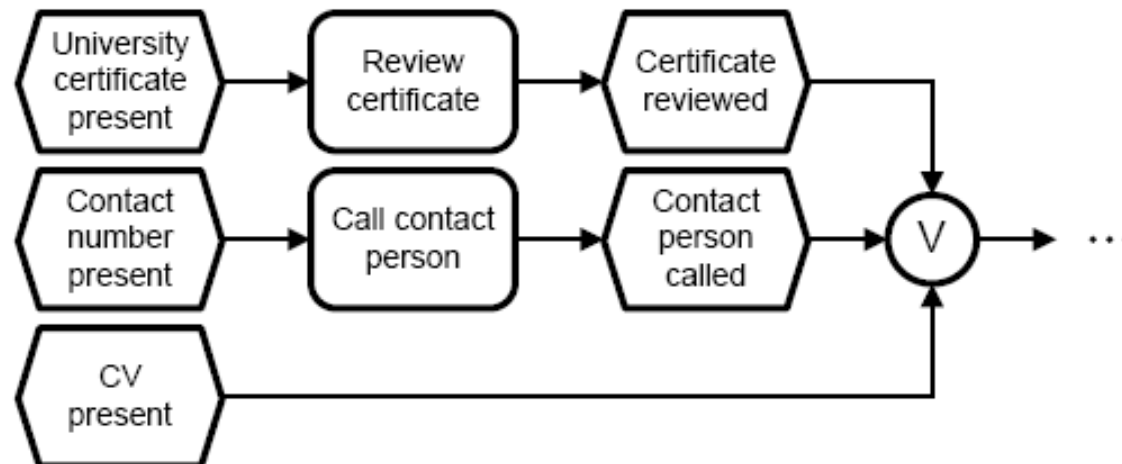
CASU: Activation

- A-1 Initial State
- **A-2 All Start Places**
 - Initial state is determined by start places
 - Each is activated during the process instantiation
- A-3 True Conditions
- A-4 Occurred Events
- A-5 Occurred Events and Conditions



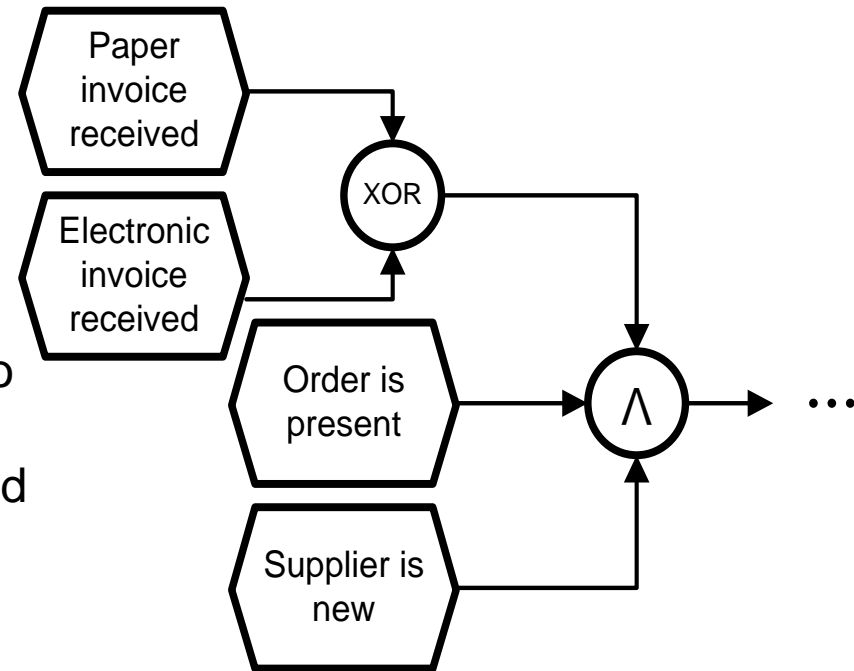
CASU: Activation

- A-1 Initial State
- A-2 All Start Places
- **A-3 True Conditions**
 - Start conditions are evaluated
 - Insofar true, the process element is activated
 - Logical coupling between activation and instantiation
- A-4 Occurred Events
- A-5 Occurred Events and Conditions



CASU: Activation

- A-1 Initial State
- A-2 All Start Places
- A-3 True Conditions
- A-4 Occurred Events
- **A-5 Occurred Events and Conditions**
 - In this case all occurred events map to activated control threads (A-4). Additionally, branches can be activated if start conditions yield true at instantiation time.



CASU: Subscription

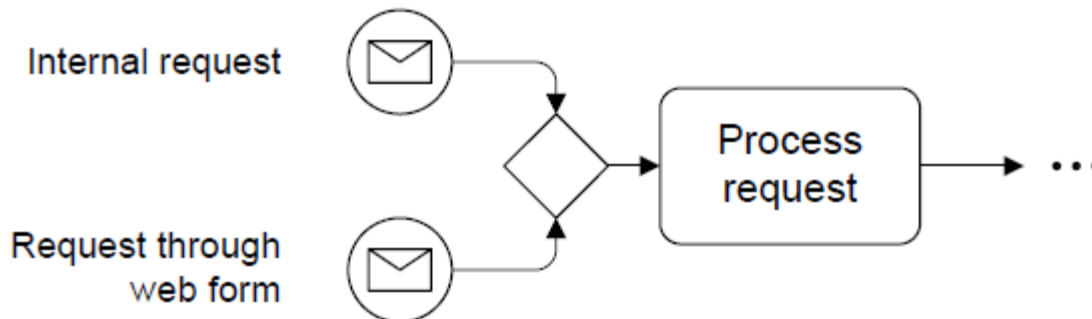
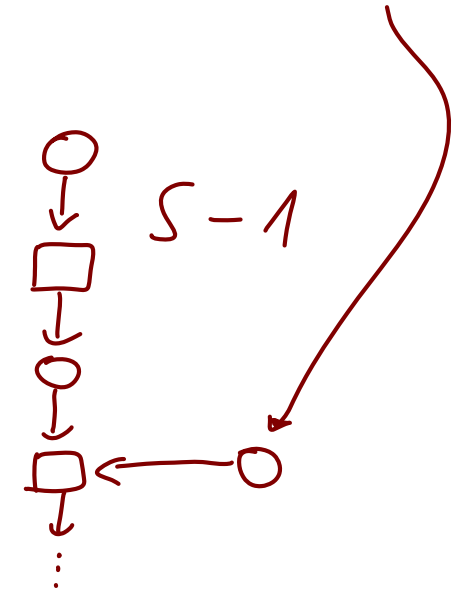
- **S-1 All Subscriptions**

- Instance subscribes for all entry points that have not yet occurred (for the corresponding events)

- **S-2 No Subscriptions**

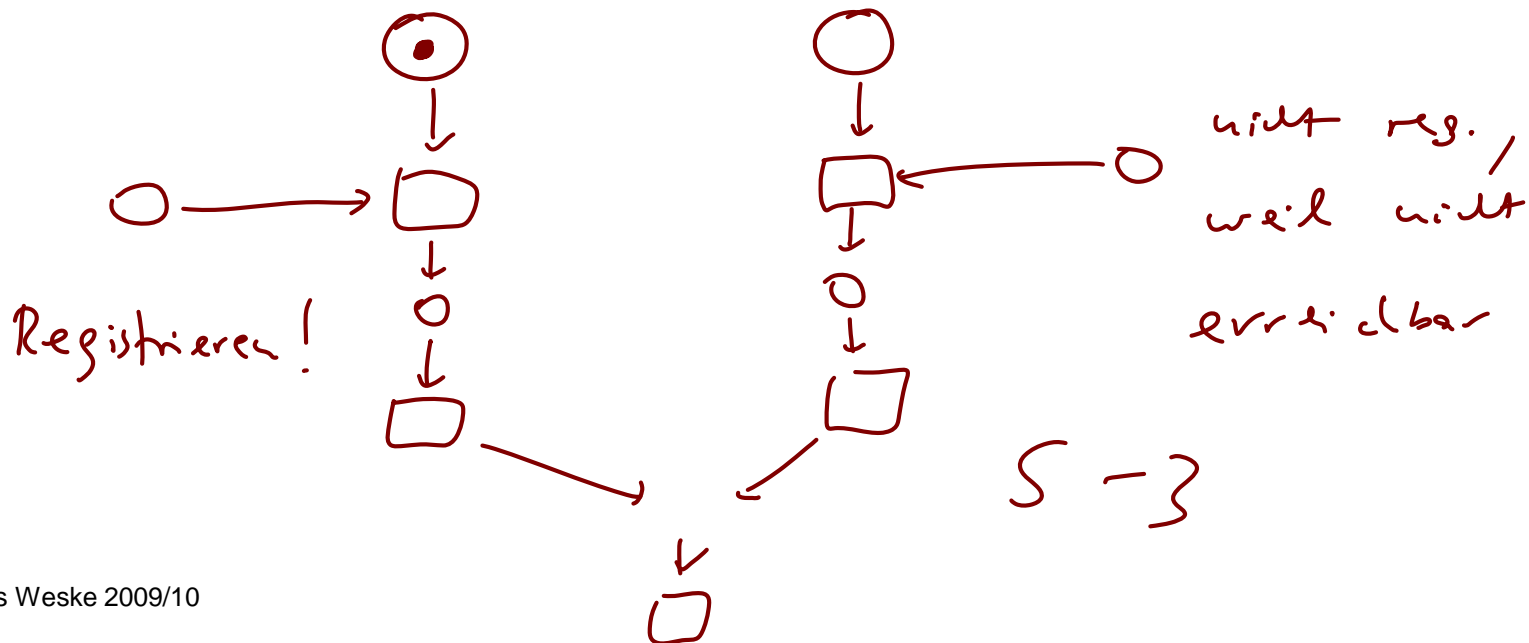
- Instance does not subscribe for any of the not occurring entry points

- **S-3 Reachable Subscriptions**



CASU: Subscription

- S-1 All Subscriptions
- S-2 No Subscriptions
- **S-3 Reachable Subscriptions**
 - Only those event subscriptions are activated that might be required later to complete the process instance properly.



CASU: Unsubscription

- U-1 Until Consumption
 - All subscriptions for existing entry points are kept
 - The process can not terminate without the occurrence of these events
- U-2 Until Termination
- U-3 Timer-based
- U-4 Event-based
- U-5 Proper Completion

```

<flow>
  <sequence><receive name="rcvRFIDNotification" ..
    createInstance="yes">
    <correlations><correlation set="cont"
      initiate="join"/></correlations>
    </receive>..</sequence>
  <sequence><receive name="rcvRoutingInfo" .. createInstance="yes">
    <correlations><correlation set="cont"
      initiate="join"/></correlations>
    </receive>..</sequence>
</flow>

```

CASU: Unsubscription

- U-1 Until Consumption
- **U-2 Until Termination**
 - The subscriptions for entry points to be kept until the process instance terminates
- U-3 Timer-based
- U-4 Event-based
- U-5 Proper Completion

```
<flow>
```

```
<receive name="rcvMsg1" .. createInstance="yes" />
```

```
<sequence><receive name="rcvMsg2" .. createInstance="yes" />
```

```
.. <exit/>
```

```
</sequence>
```

```
</flow>
```

CASU: Unsubscription

- U-1 Until Consumption
- U-2 Until Termination
- **U-3 Timer-based**
 - After a certain period of time, the registrations for entry points are canceled
- U-4 Event-based
- U-5 Proper Completion

```

<flow>
  <receive name="rcvMsg1" .. createInstance="yes" />
  <pick createInstance="yes">
    <onMessage name="rcvMsg2" .. >
    .. </onMessage>
    <onAlarm name="timeout" .. >
    .. </onAlarm>
  </pick> ..
</flow>

```

CASU: Unsubscription

- U-1 Until Consumption
- U-2 Until Termination
- U-3 Timer-based
- **U-4 Event-based**
 - After one of several exclusive Start events occurred, the subscriptions for the other entry points are canceled
- U-5 Proper Completion

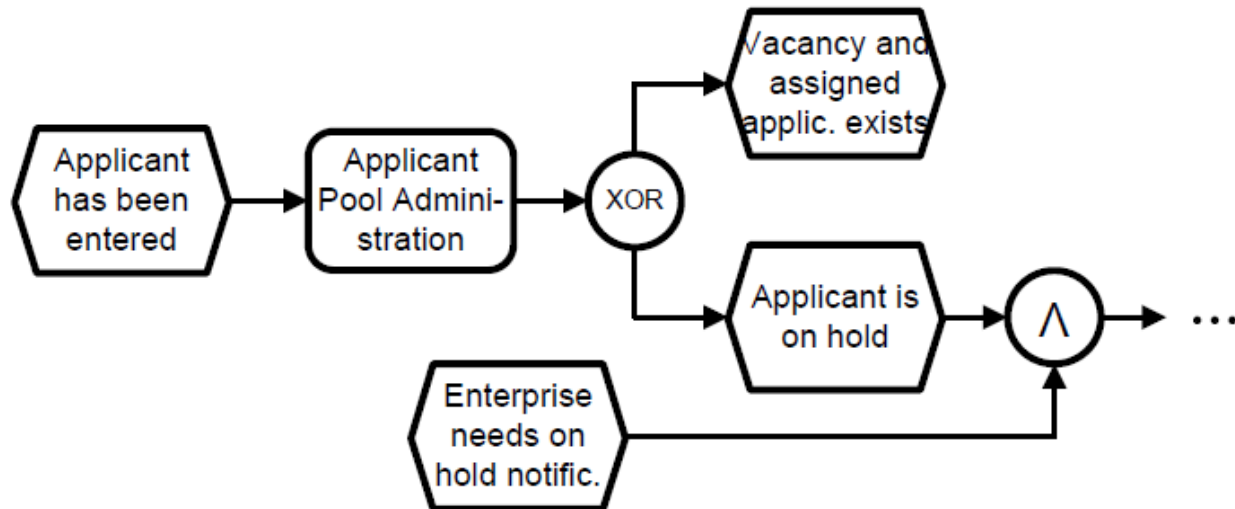
```

<flow>
  <receive name="rcvInvoice" .. createInstance="yes" />
  <pick createInstance="yes">
    <onMessage name="rcvPaperDeliveryNotification" .. >
      .. </onMessage>
    <onMessage name="rcvElectronicDeliveryNotification" .. >
      .. </onMessage>
  </pick> ..
</flow>

```

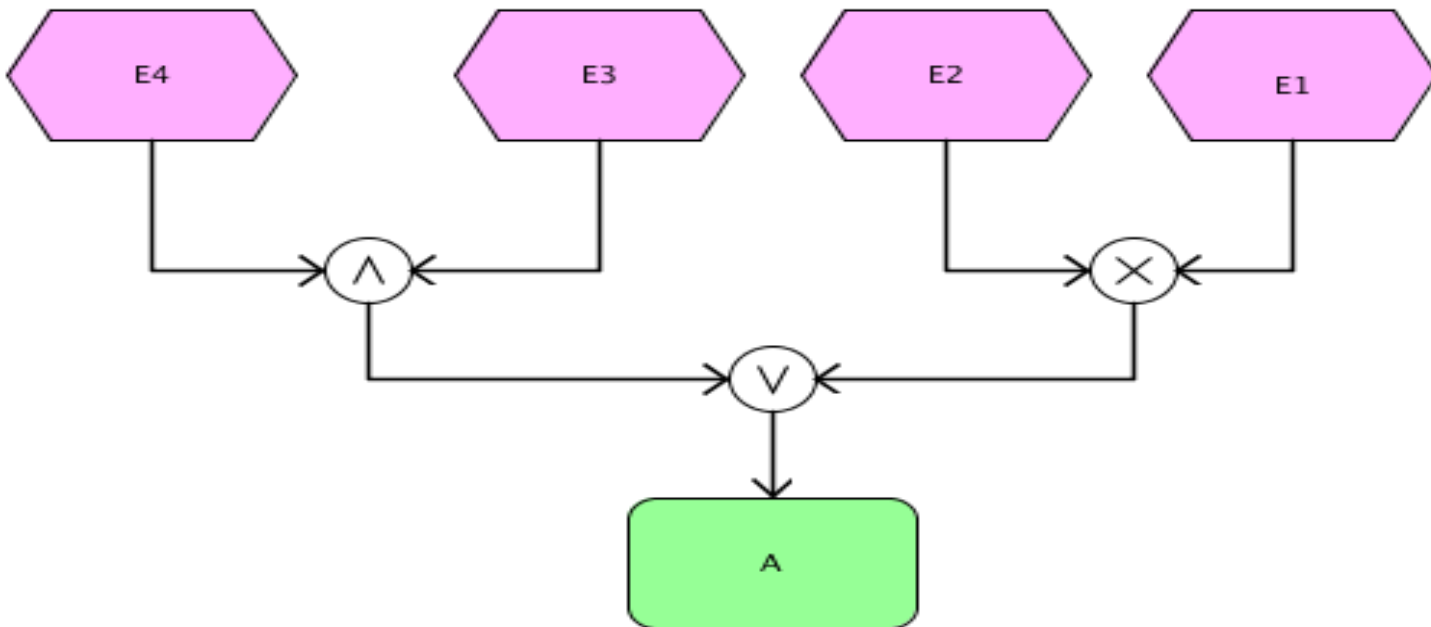
CASU: Unsubscription

- U-1 Until Consumption
- U-2 Until Termination
- U-3 Timer-based
- U-4 Event-based
- **U-5 Proper Completion**
 - Registration is discarded once the process reaches a valid end state without the event



Example:

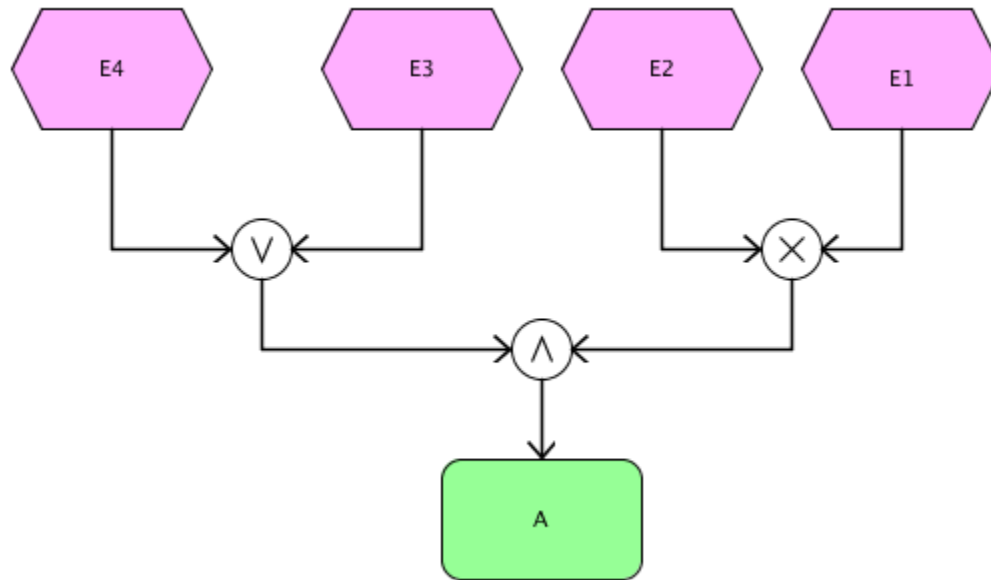
- $t_1 : e_4(1)$, $t_2 : e_2(1)$, $t_3 : e_3(2)$, $t_4 : e_4(2)$, $t_5 : e_1(1)$, $t_6 : e_3(1)$, $t_7 : e_2(3)$, $t_8 : e_3(3)$, $t_9 : e_4(3)$, $t_{10} : e_1(3)$



t1 : e4(1), t2 : e2(1), t3 : e3(2), t4 : e4(2), t5 : e1(1), t6 : e3(1), t7 : e2(3), t8 : e3(3), t9 : e4(3), t10 : e1(3)

↳ Subscription: All events

↳ Un-subscription: upon consumption



t1 : e4(1), t2 : e2(1), t3 : e3(2), t4 : e4(2), t5 : e1(1), t6 : e3(1), t7 : e2(3), t8 : e3(3), t9 : e4(3), t10 : e1(3)

↳ Subscription: Missing events

↳ Un-subscription: until termination

