# Verification and Model checking
# DBMA-spring 2016

Håkon Normann

May 9, 2016

# Overview

- Soundness

# Overview

- Soundness
- Deadlock/livelock

# Overview

- Soundness
- Deadlock/livelock
- State machines/Transition systems

# Overview

- Soundness
- Deadlock/livelock
- State machines/Transition systems
- Finding deadlocks/livelocks in Transition systems.

# Need something fitting here

- Soundness
  1. Option to complete

# Need something fitting here

- Soundness
  1. Option to complete
  2. Proper completion

# Need something fitting here

- Soundness
  1. Option to complete
  2. Proper completion
  3. No dead activities.

# Need something fitting here

- Soundness
  1. Option to complete
  2. Proper completion
  3. No dead activities.
- Data Flow Correctness
  1. No missing data

# Need something fitting here

- Soundness
  1. Option to complete
  2. Proper completion
  3. No dead activities.
- Data Flow Correctness
  1. No missing data
  2. No dead activities

# Need something fitting here

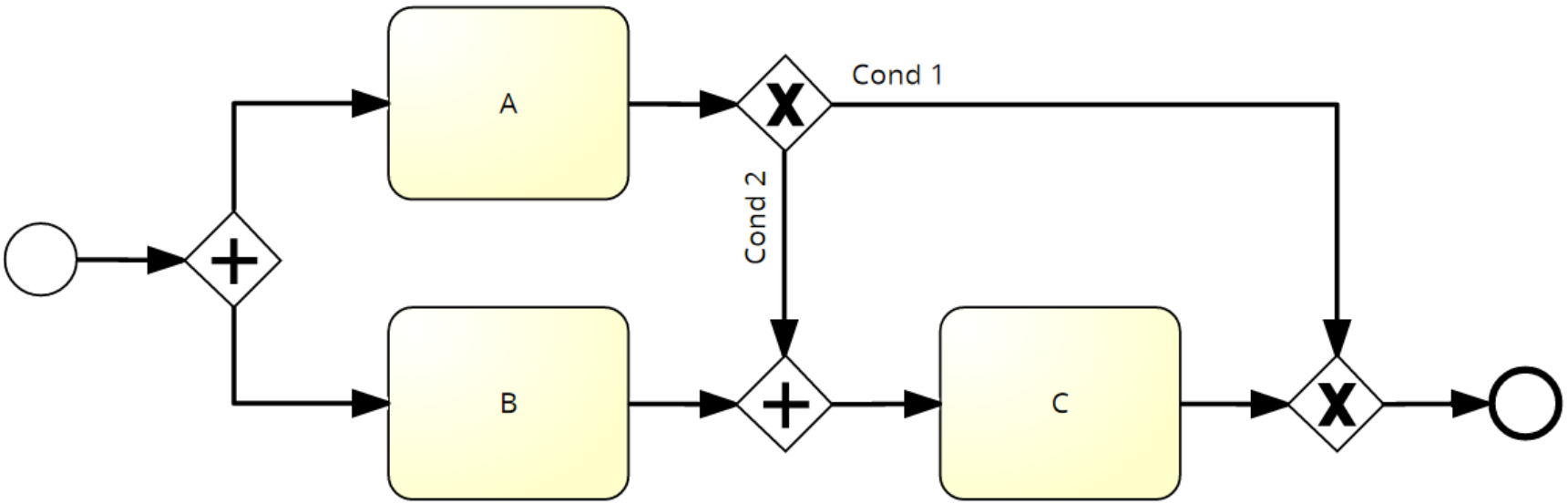
- Soundness
  1. Option to complete
  2. Proper completion
  3. No dead activities.
- Data Flow Correctness
  1. No missing data
  2. No dead activities
  3. No lost updates

# Option to Complete

A process instance, once started, can always complete.
Require that the process is free of Deadlocks and Livelocks.

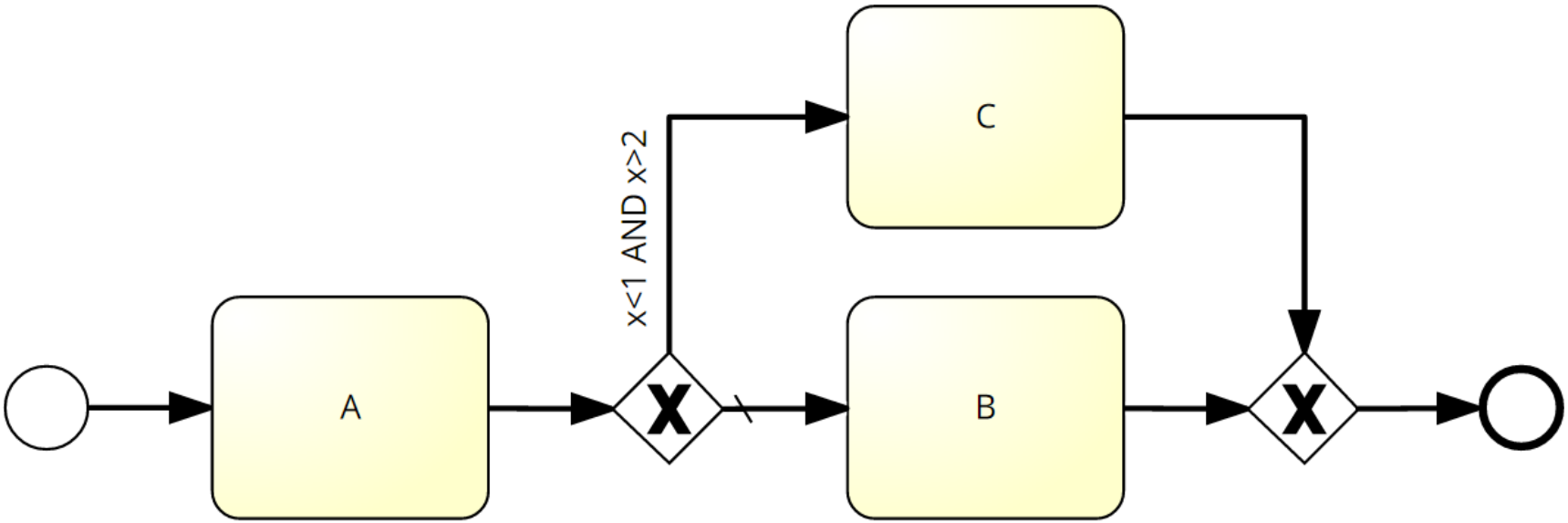
# Proper completion

When a process instance completes there is no related activity of this instance which is still running or enabled.



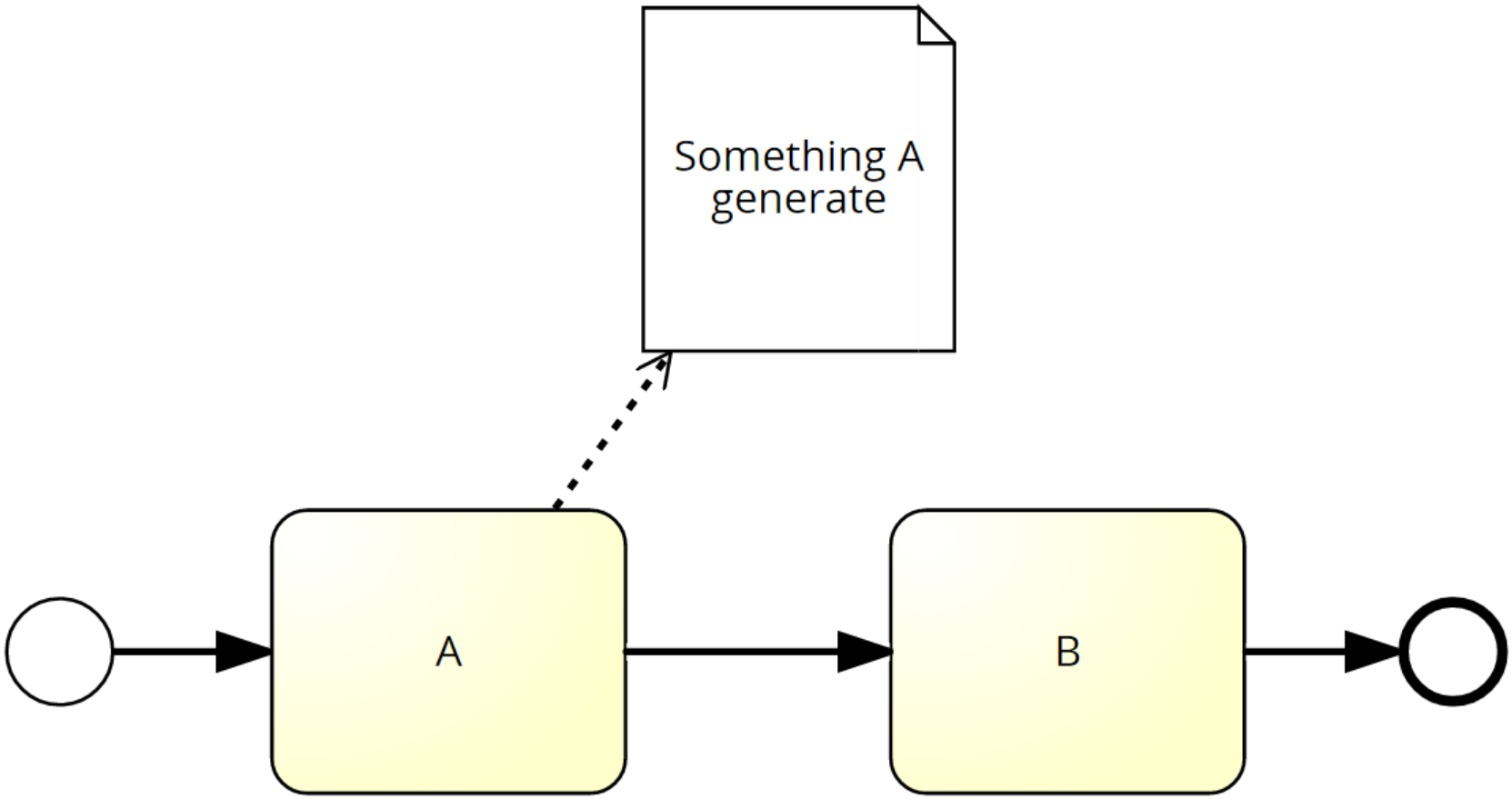
# Dead Activity

For each activity there exists at least one completed trace producible on that model and containing this activity.



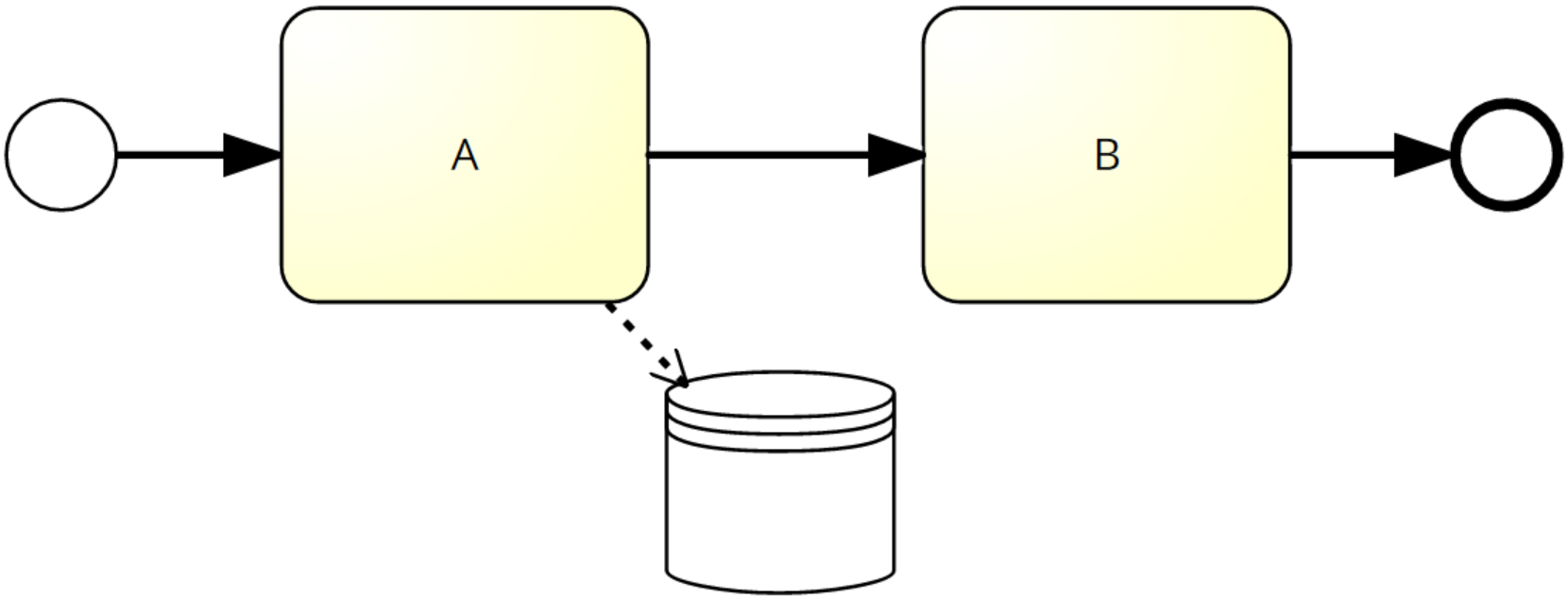
# Unnecessary Data

A data object written by an activity of process model is called unnecessary if it is not read by any subsequent activity or transition condition or passed to the outside environment via an end message.



# How to make unnecessary data not useless

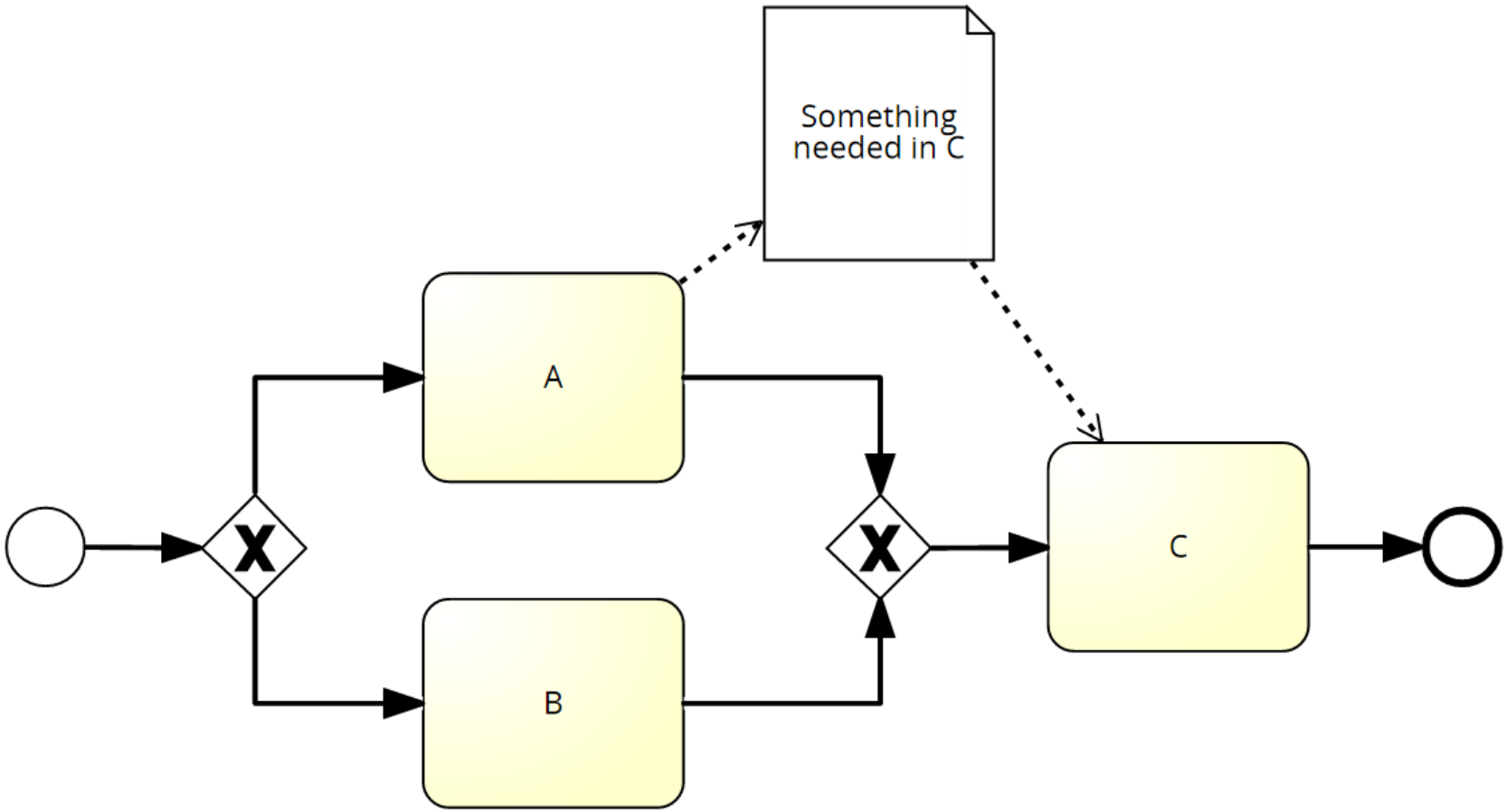
When you want data created in an instance to survive the end of the process, store the data in a data storage.




Storage for data A creates

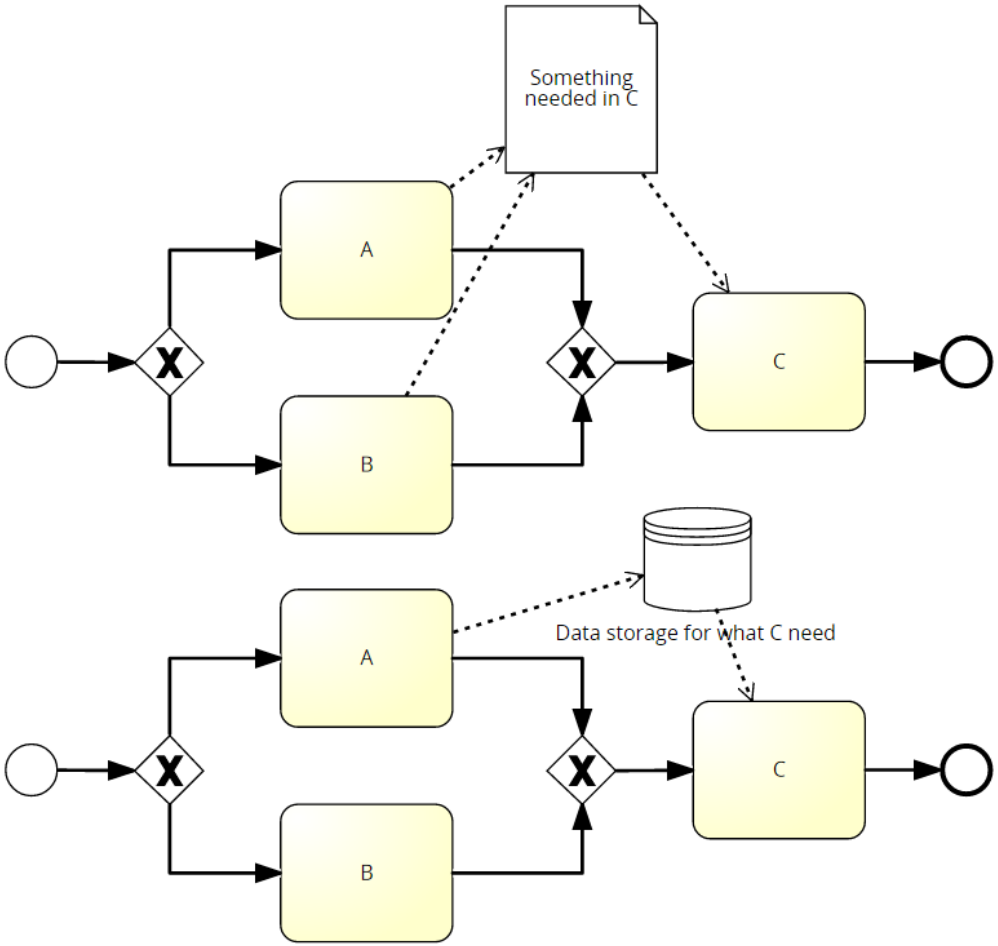
# Missing Data

The data flow schema of a process model might cause missing data at run-time if a data object exists which can be read during run-time without having been written by any preceding activity or provided by the outside environment (i.e., by a start message).



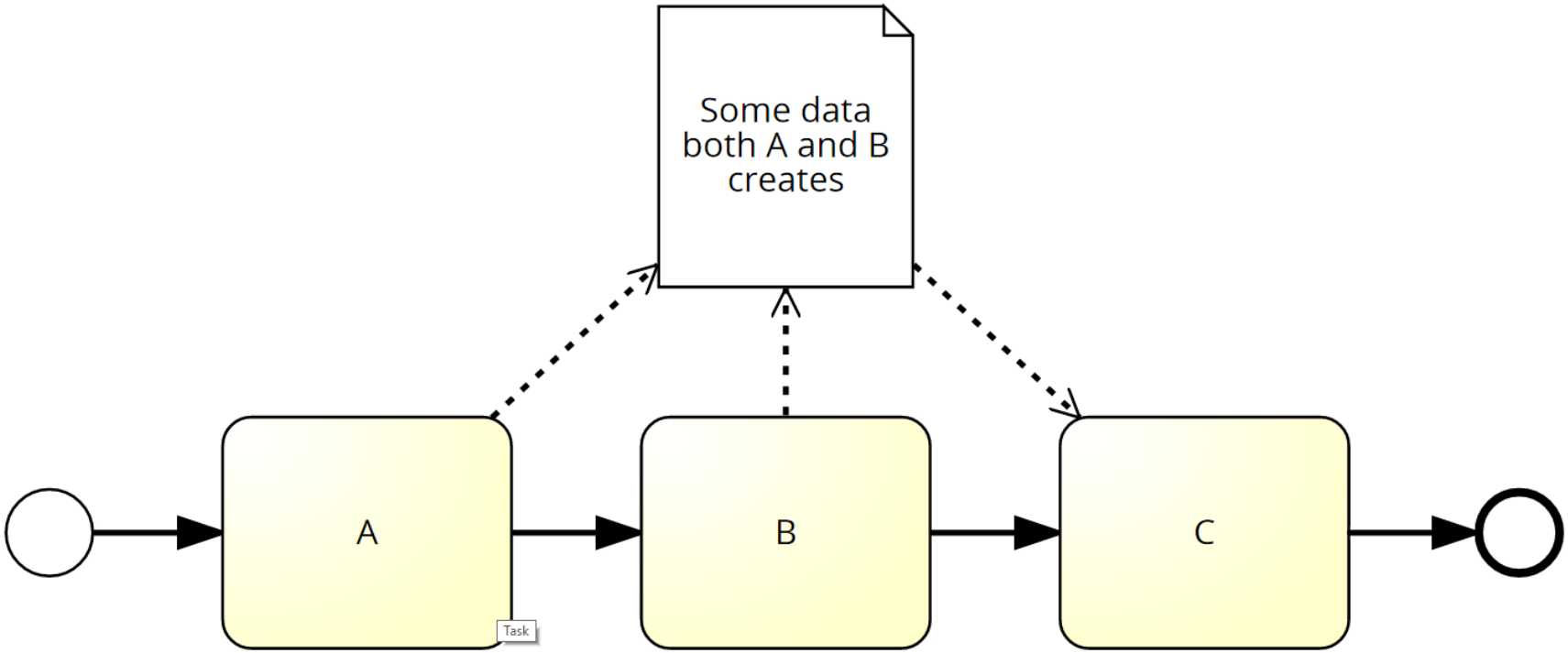
# Possible fixes to the missing data example

Either have both create the data, if the data is only living in the process instance, or if the data may live outside the process instance use a data storage.



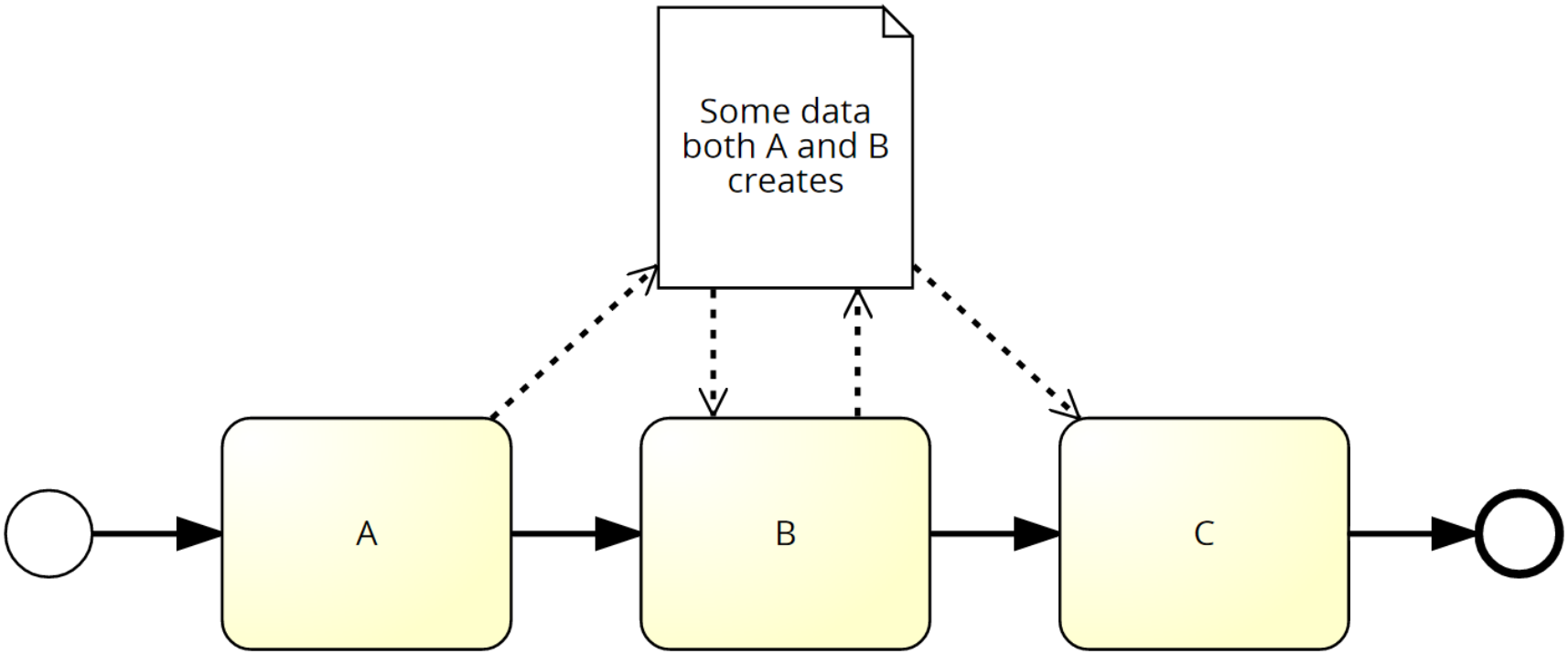
# Lost Update

The data flow schema of a process model might cause lost data at run-time if a data object, which is written by an activity, is updated by a subsequent activity, but without reading the data object in between.



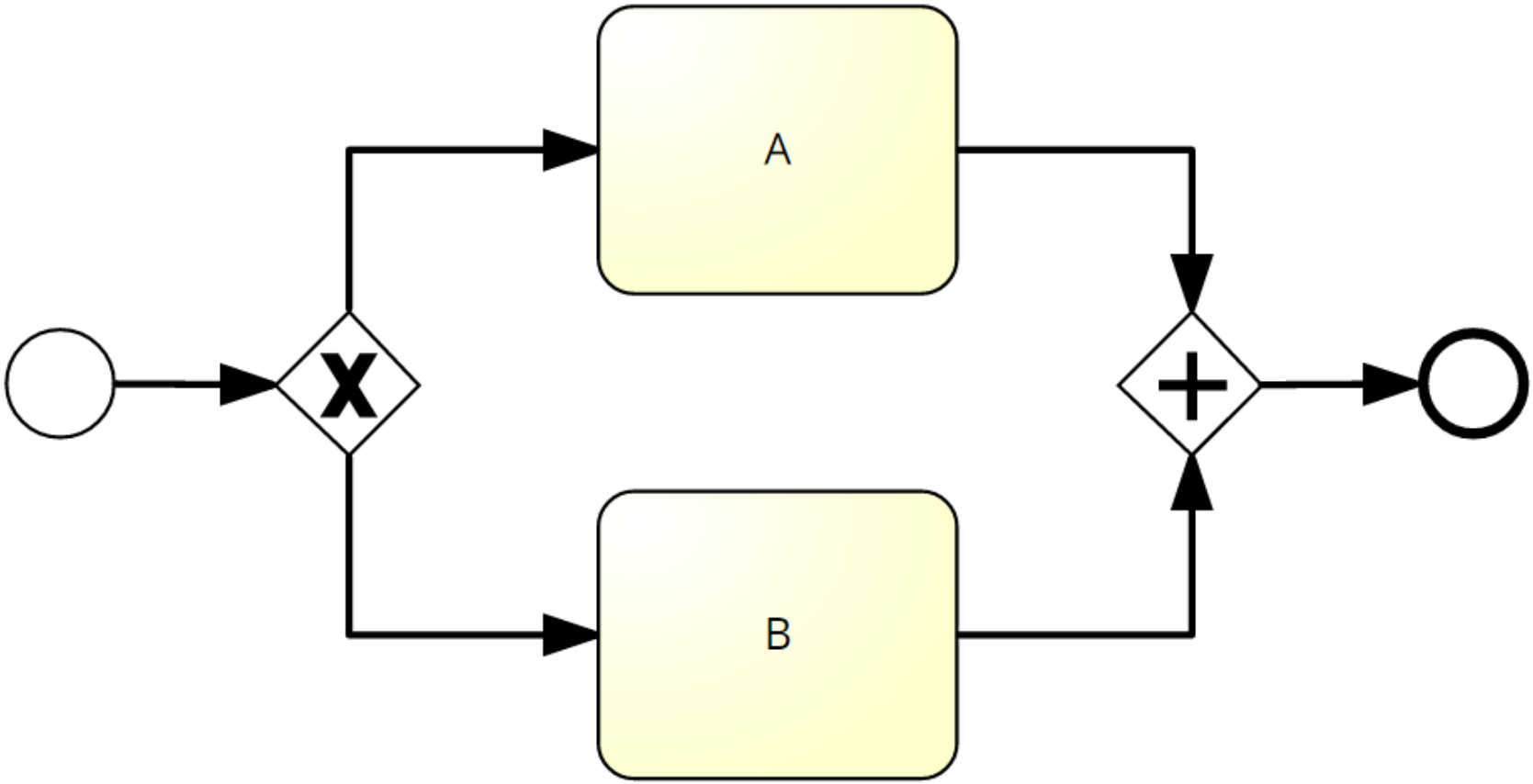
# Fix to loosing the Lost Update

Make sure that if you do not want to overwrite everything that the activity reads the data so it can update the proper parts.



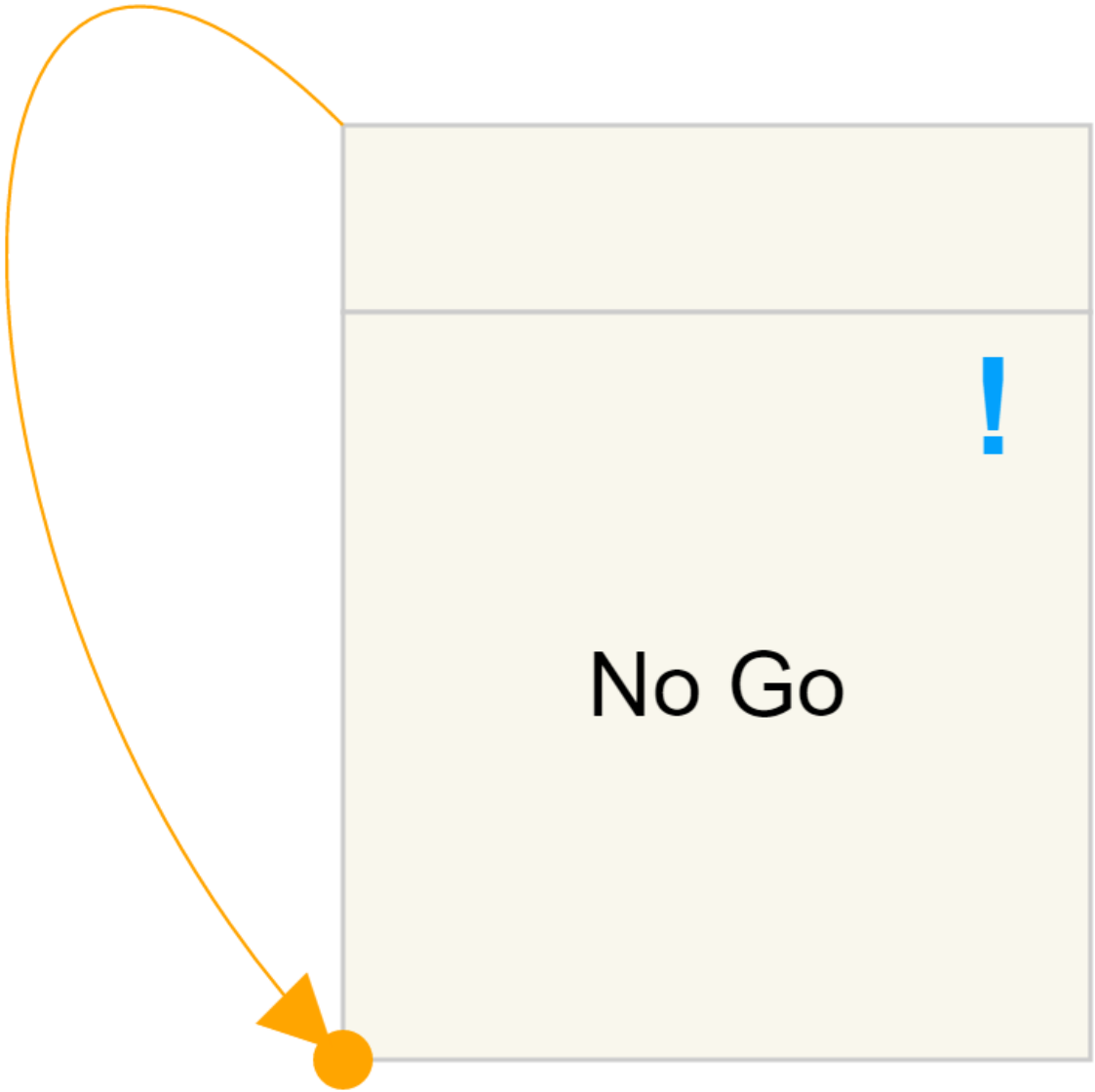
# Deadlocks

There process is in a state where no activity is enabled, and the process is not completed.



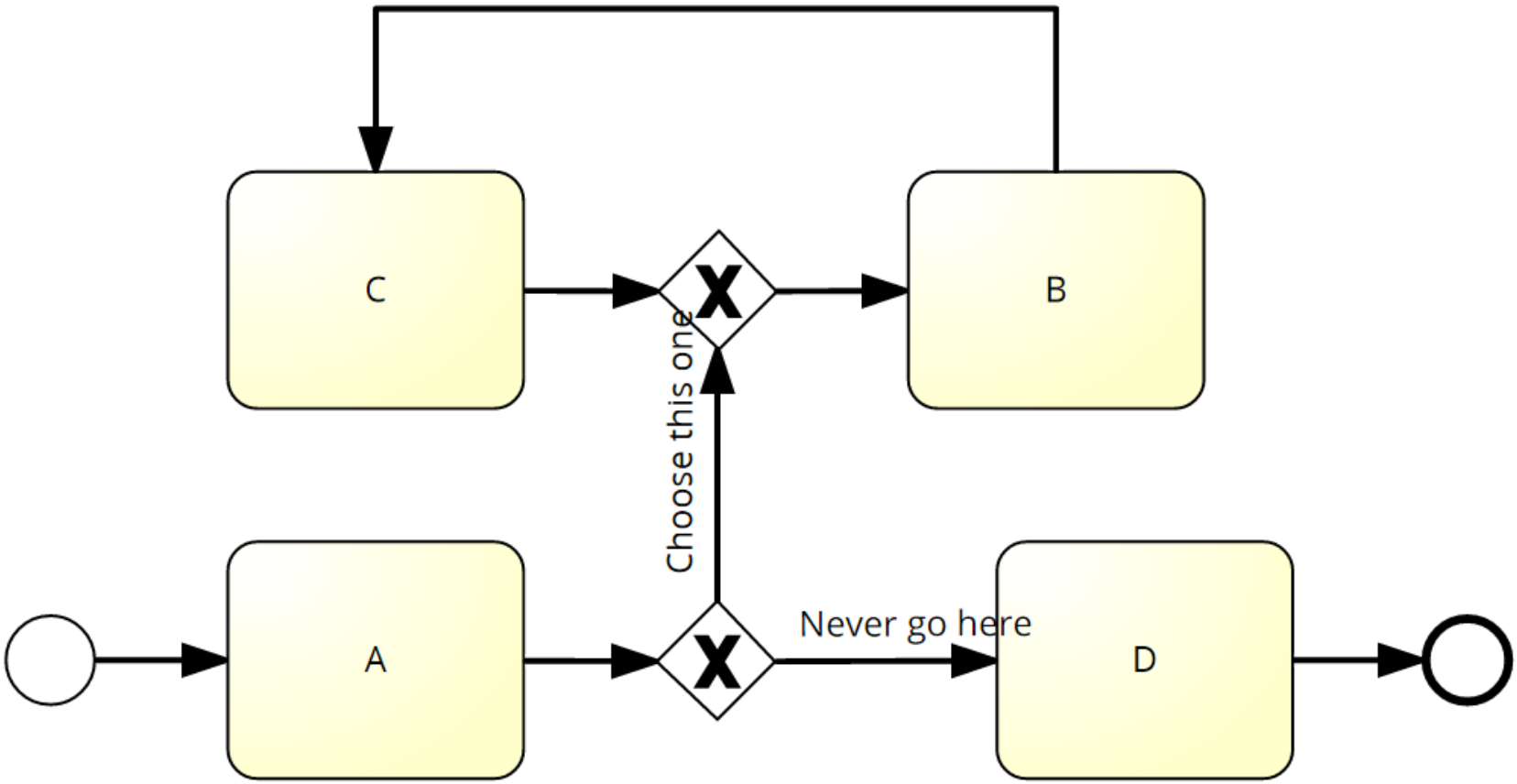
# Deadlocks in DCR's

There are a reachable state no enabled events, and there exist at least one included pending event.



# livelock

The process continuously is able to perform some activity but never reach a completed state.

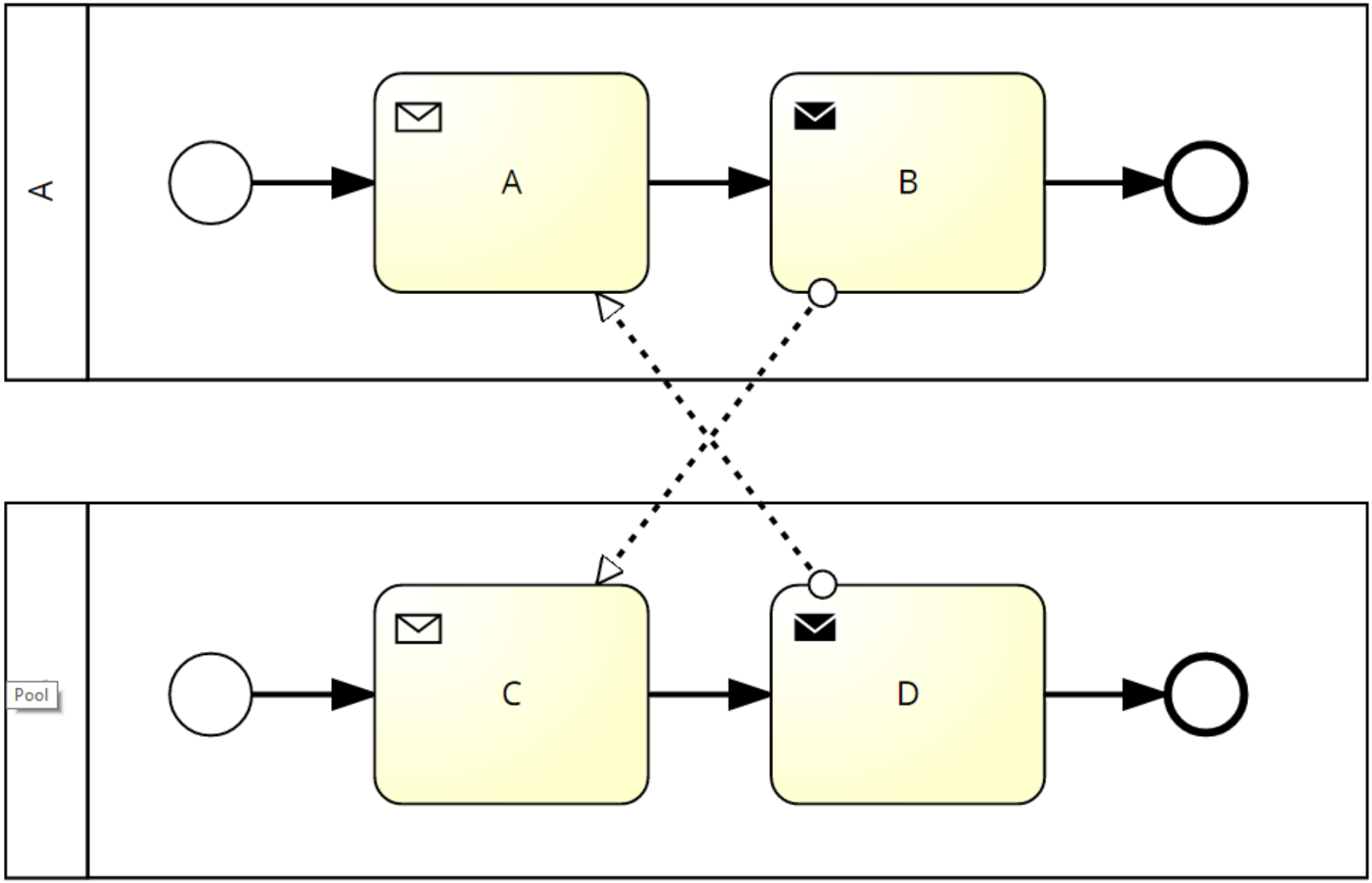


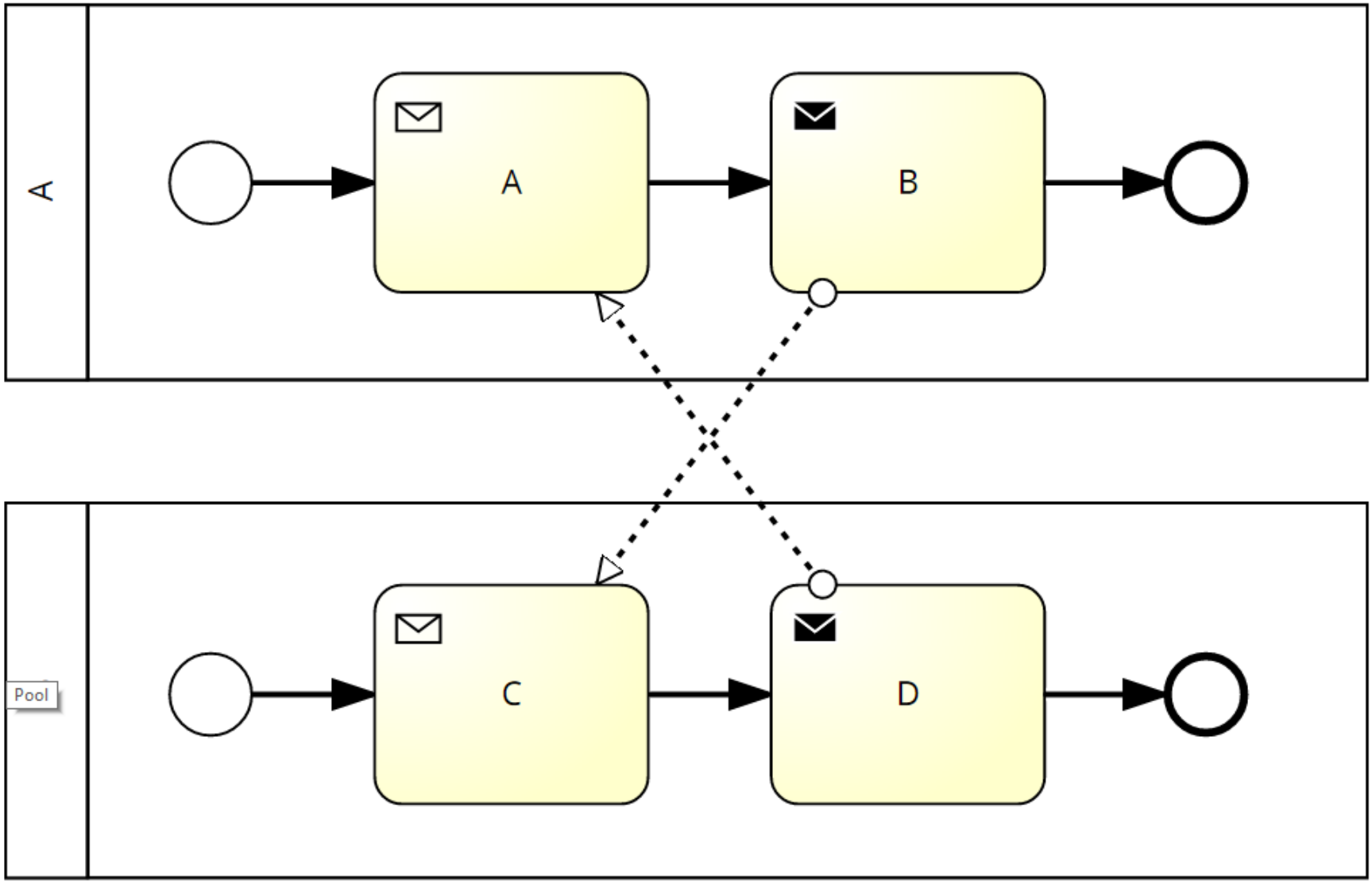
# Livelock in DCR's

For all states reachable from the current state, there exists an enabled event and at least one included pending event that never gets excluded or enabled.

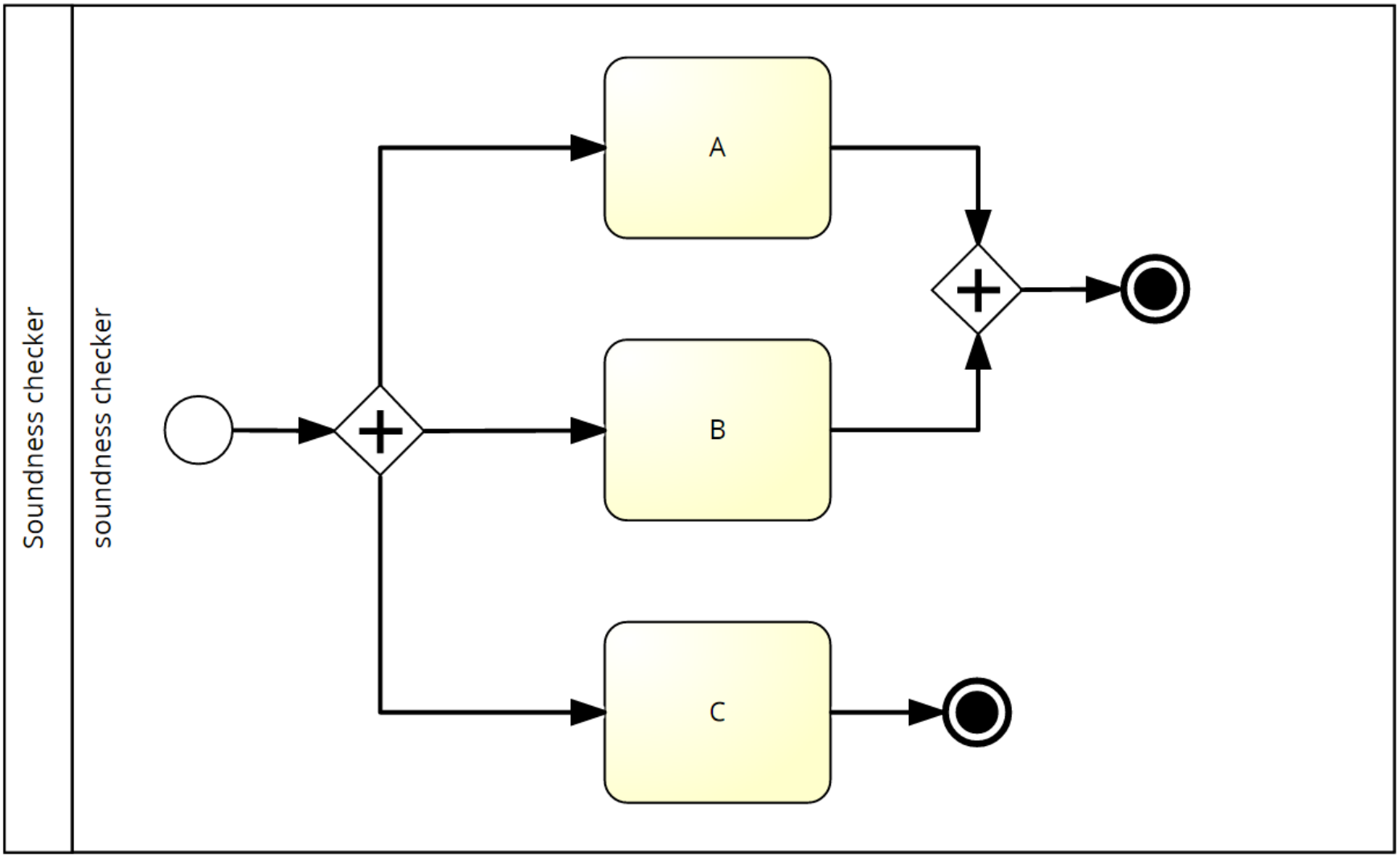
# Exercise

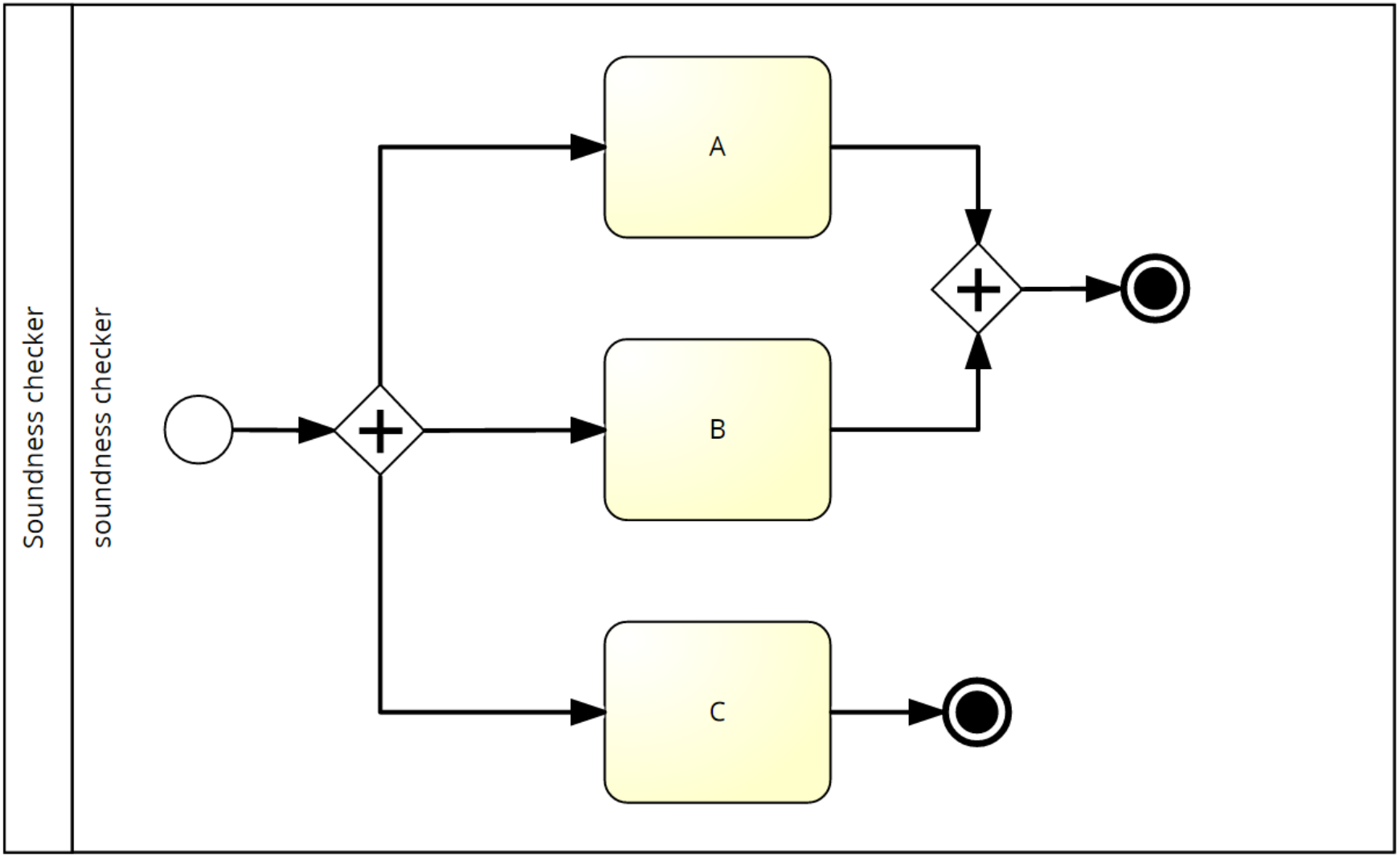
The models in the exercises file, which ones violates soundness and data flow, and how?


A
A
B
Pool
C
D

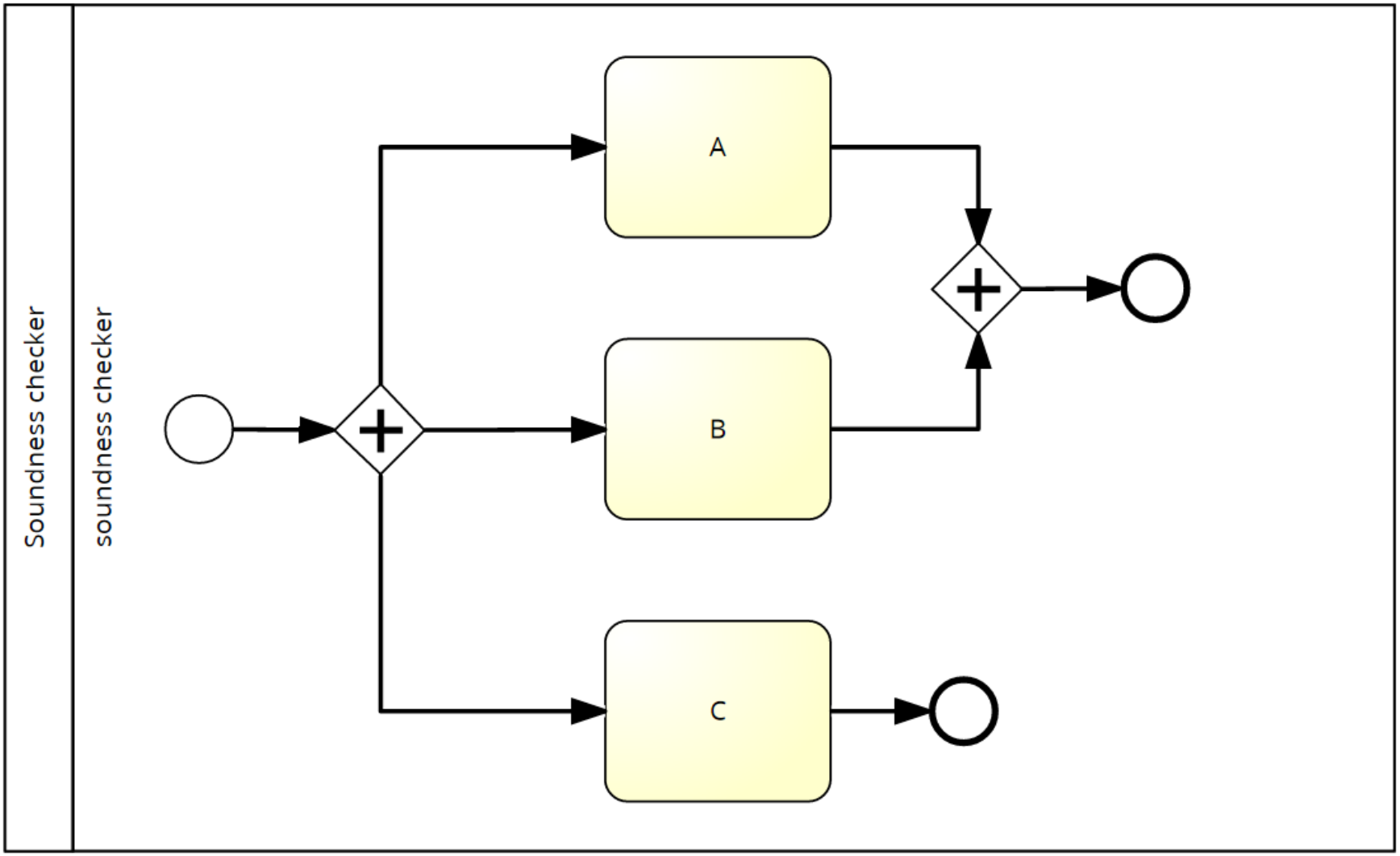


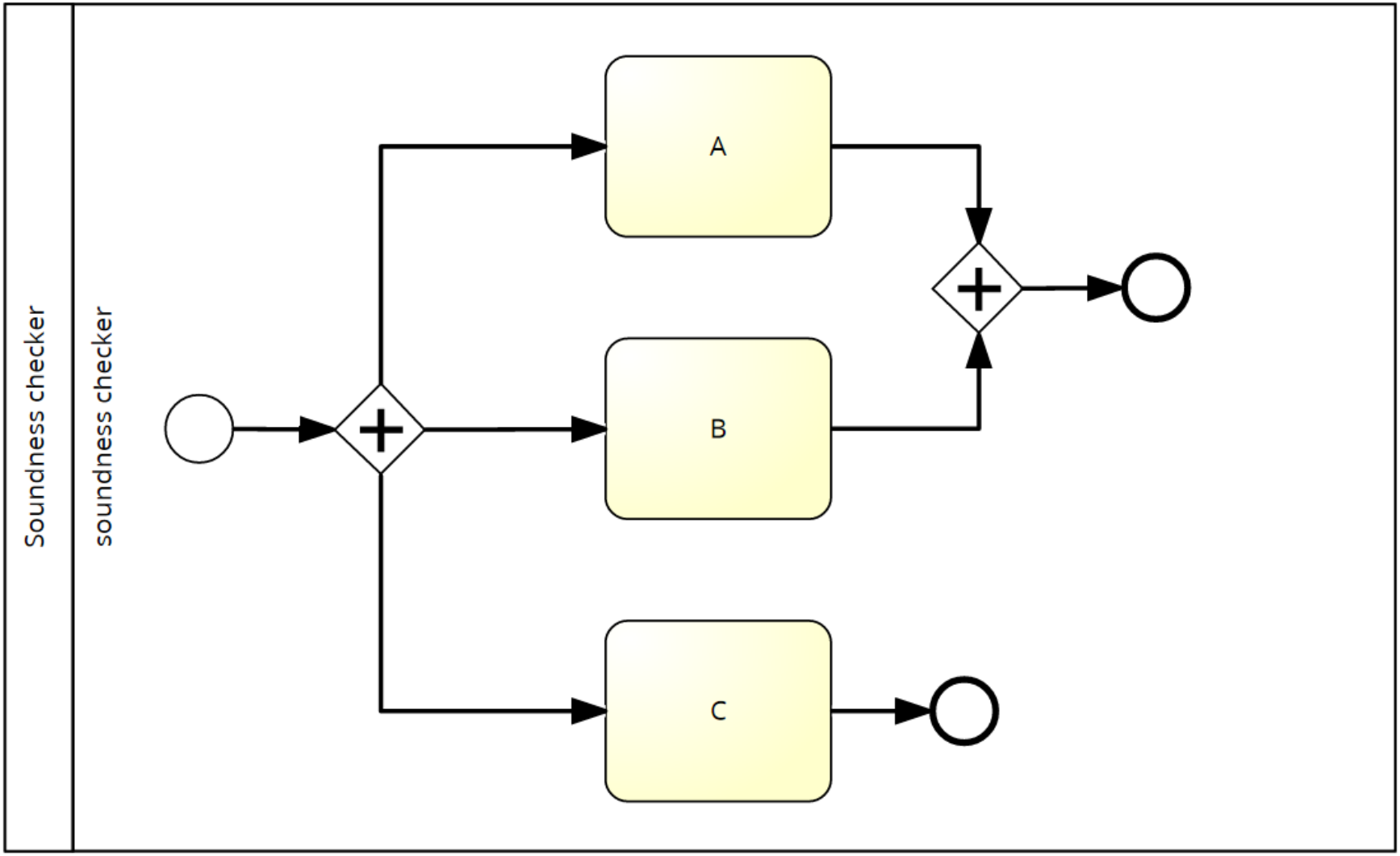

BAD: Deadlocked, Both pools have activities waiting on a message to arrive from the other pool.


Soundness checker
soundness checker
A
B
C

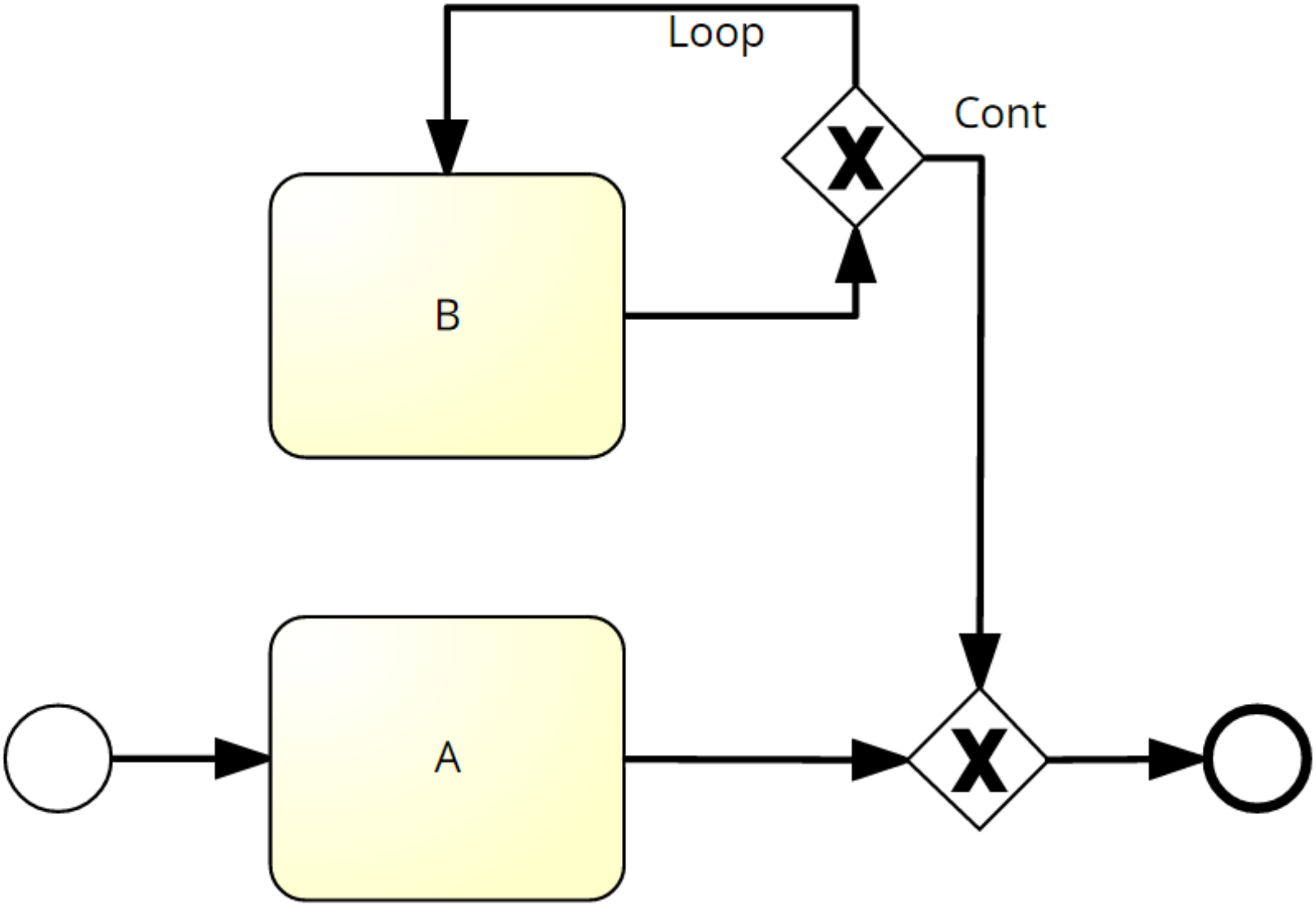


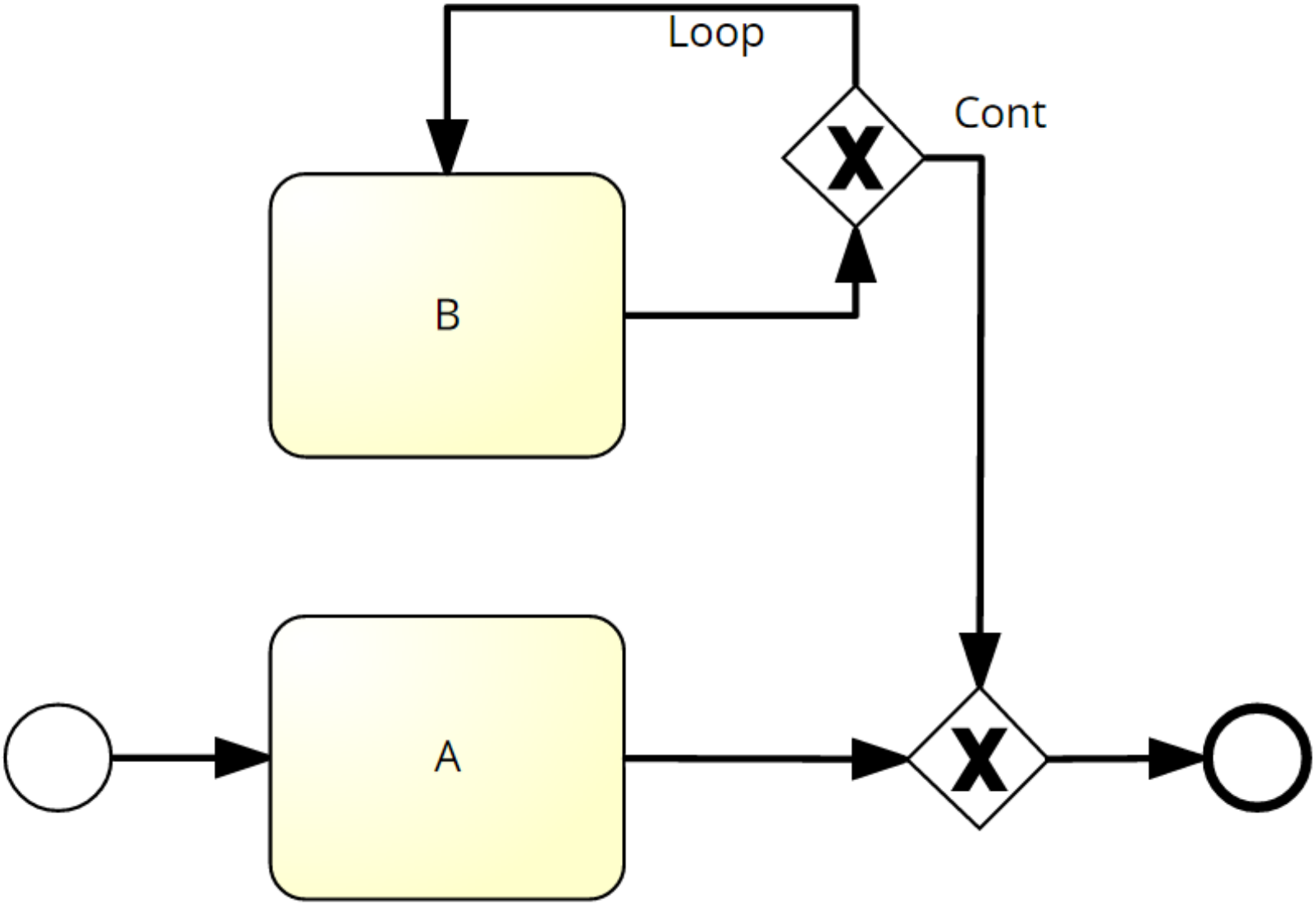

GOOD: Terminating end-events stops all execution in a process, not caring about what is currently enabled or being executed.


Soundness checker
soundness checker
A
B
C

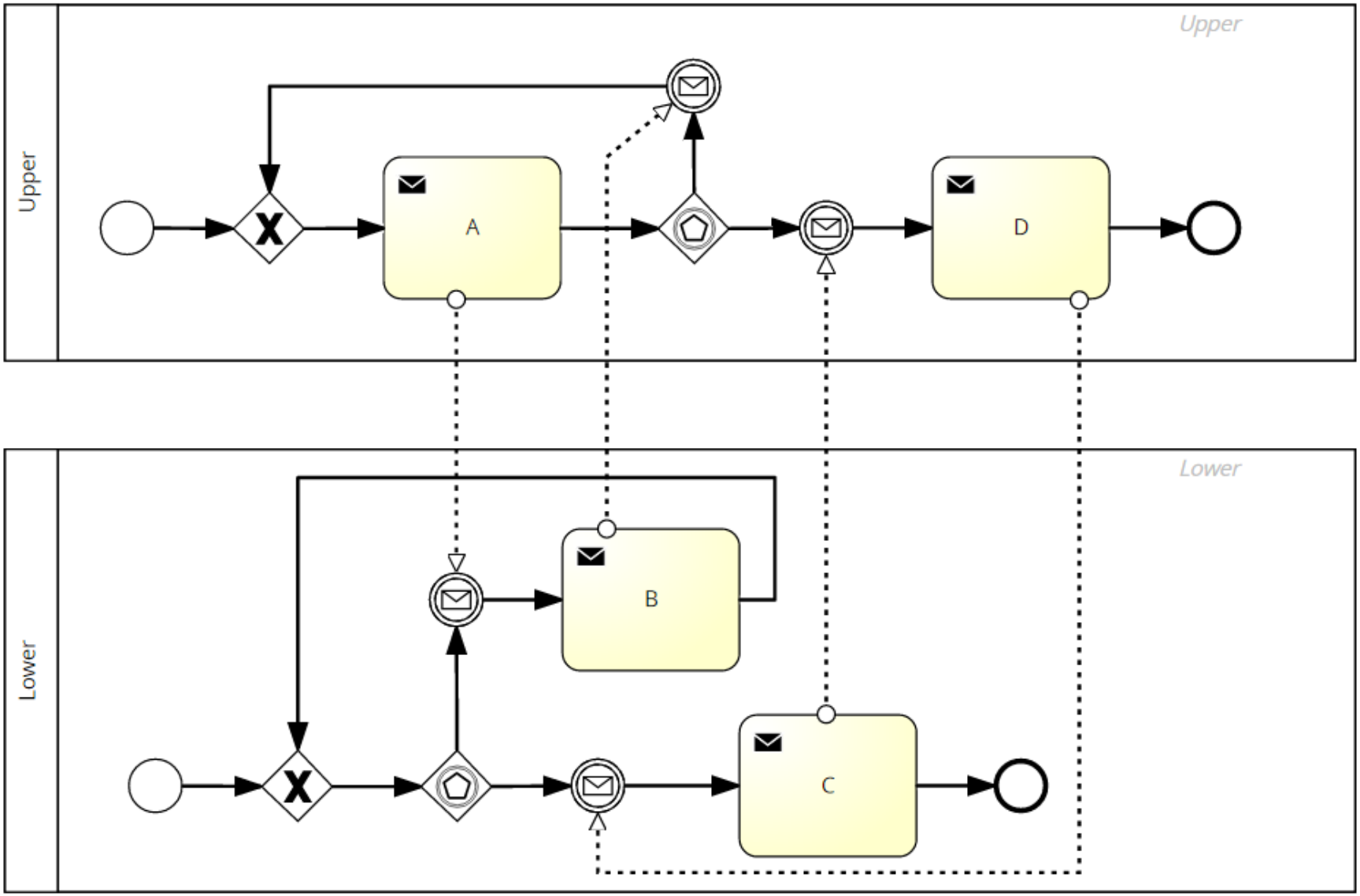


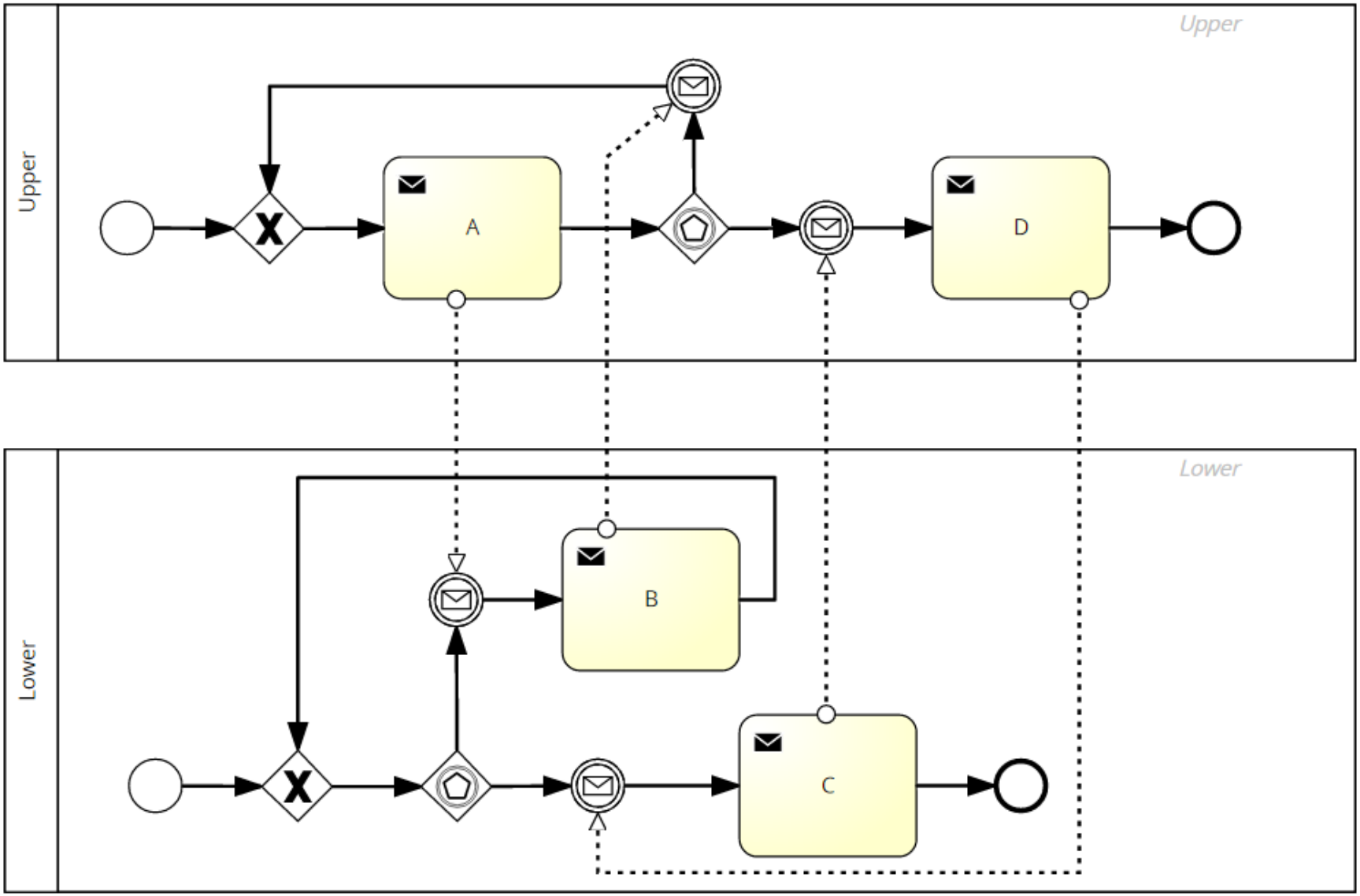

BAD: Non proper completion, once either C or A and B finishes the process gets to an end-event without the other part having finished.


Loop
Cont
B
A

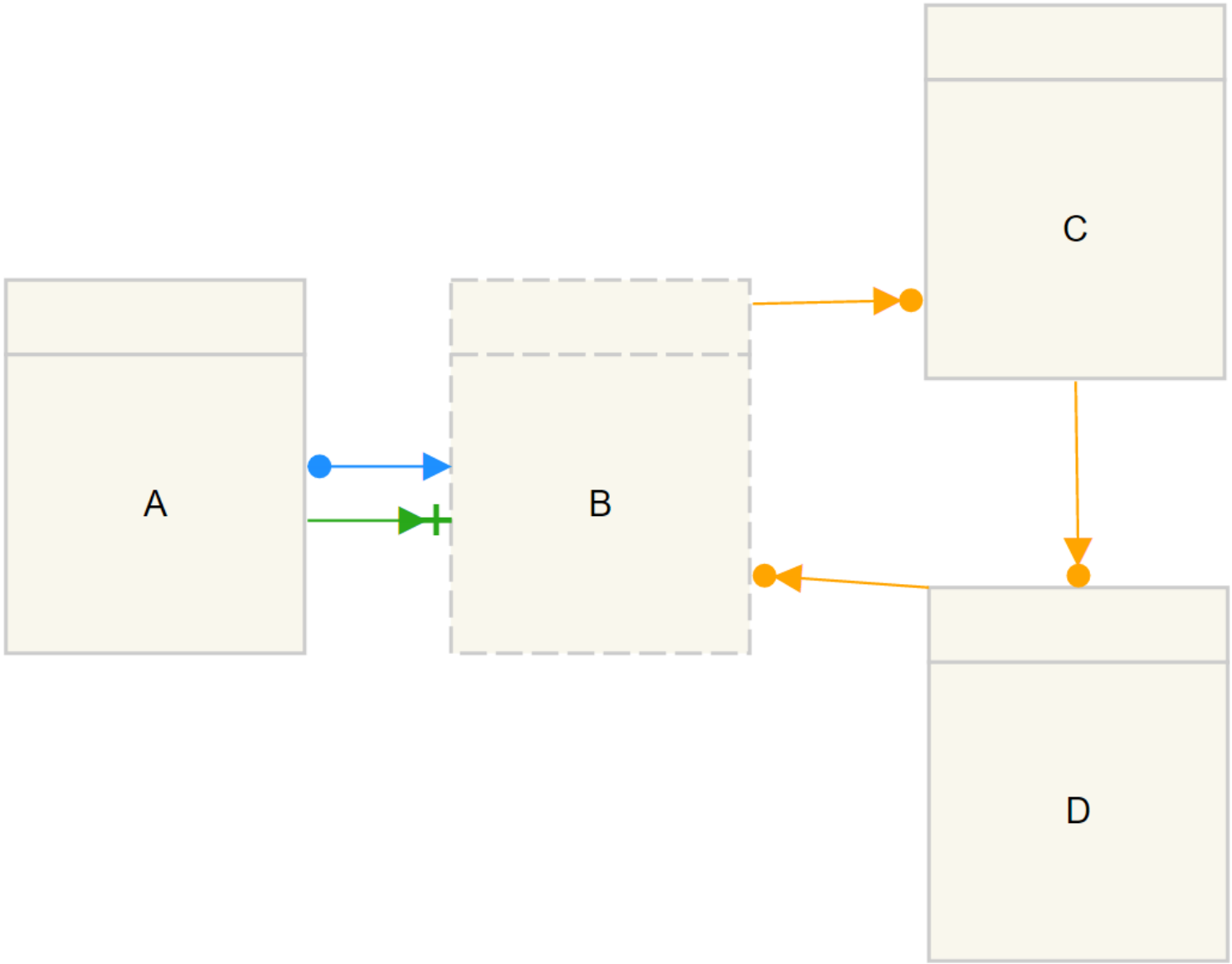


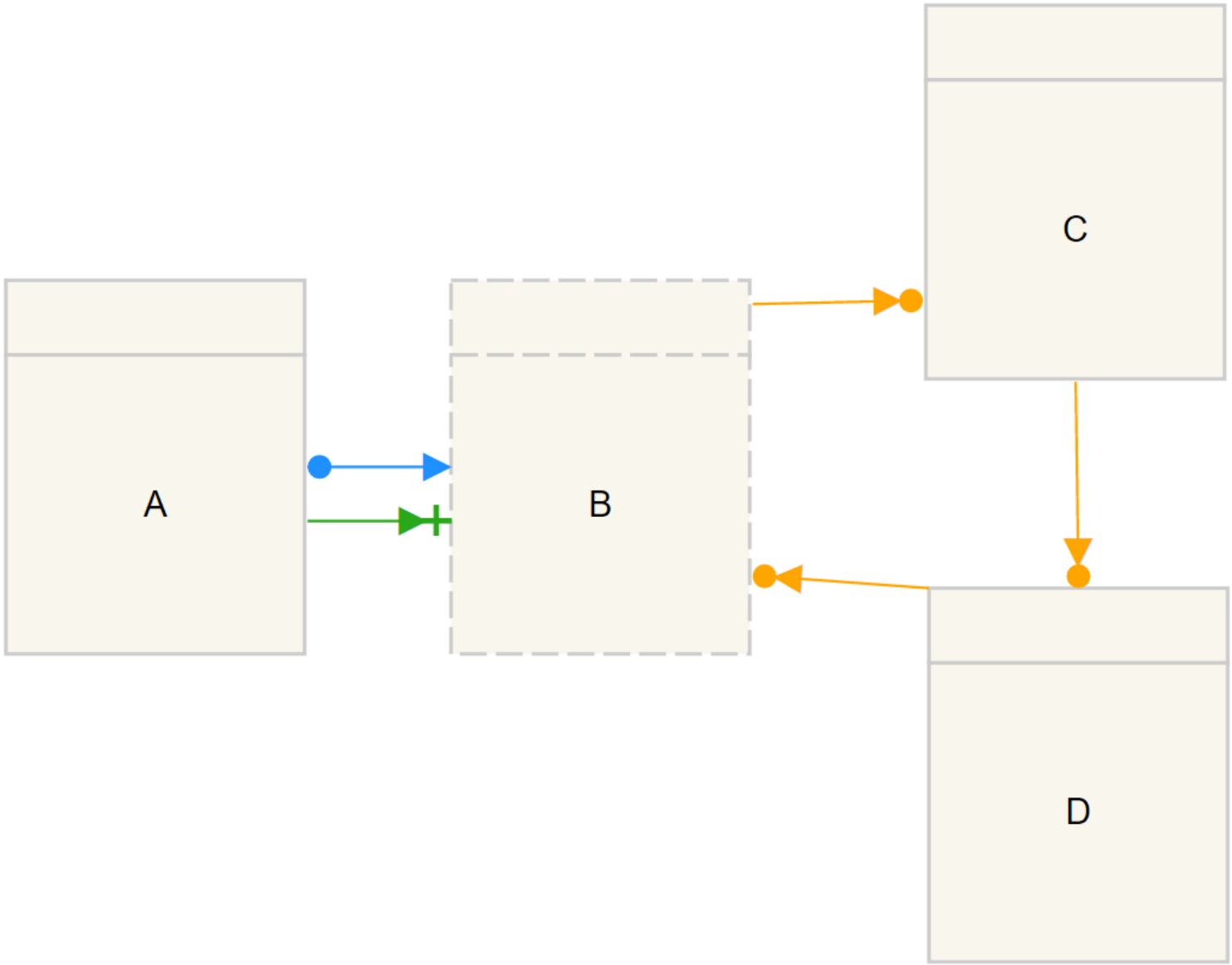

BAD: Dead activity, B can never be reached.


Upper
Upper
A
D
Lower
Lower
B
C

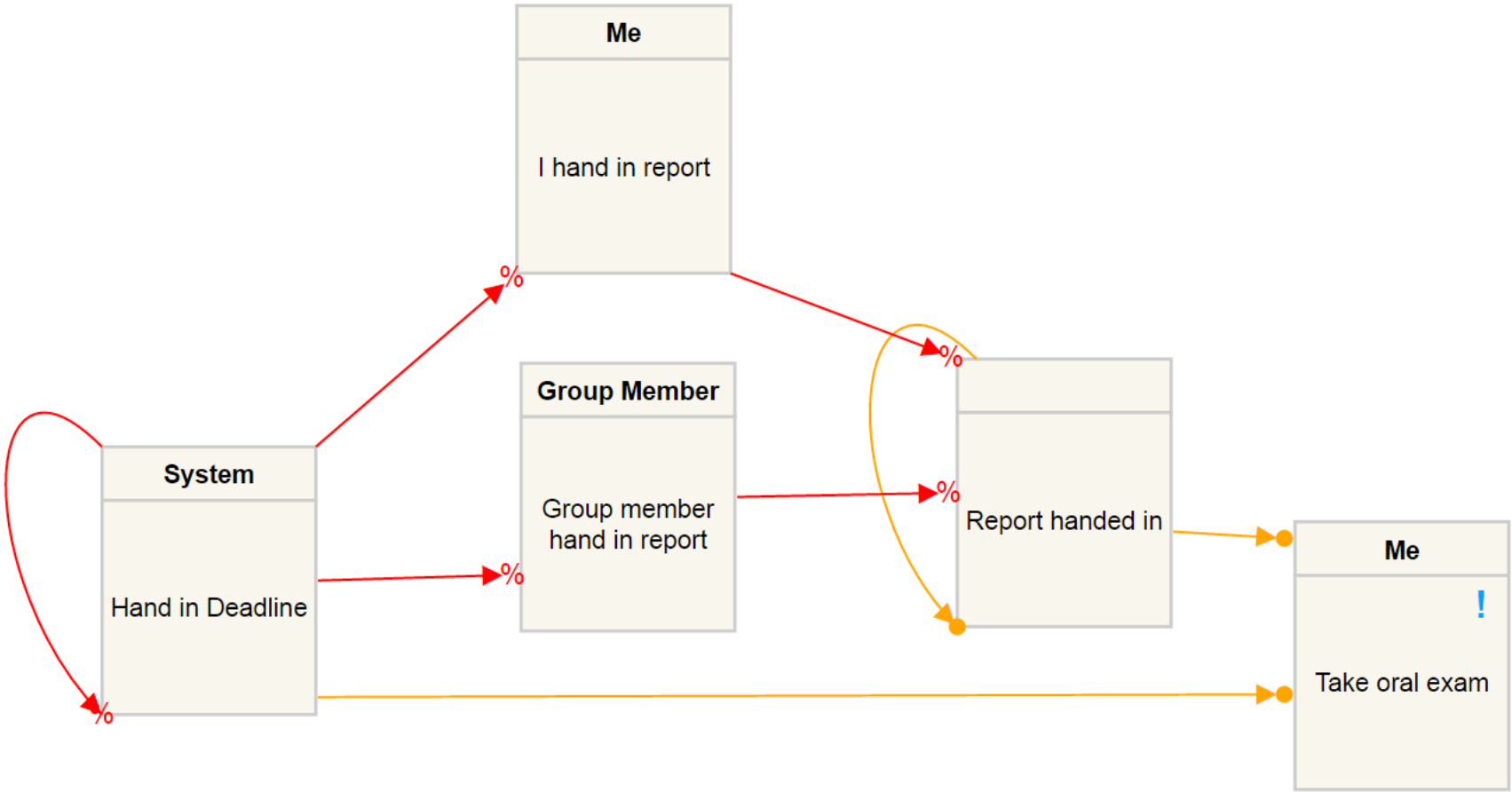


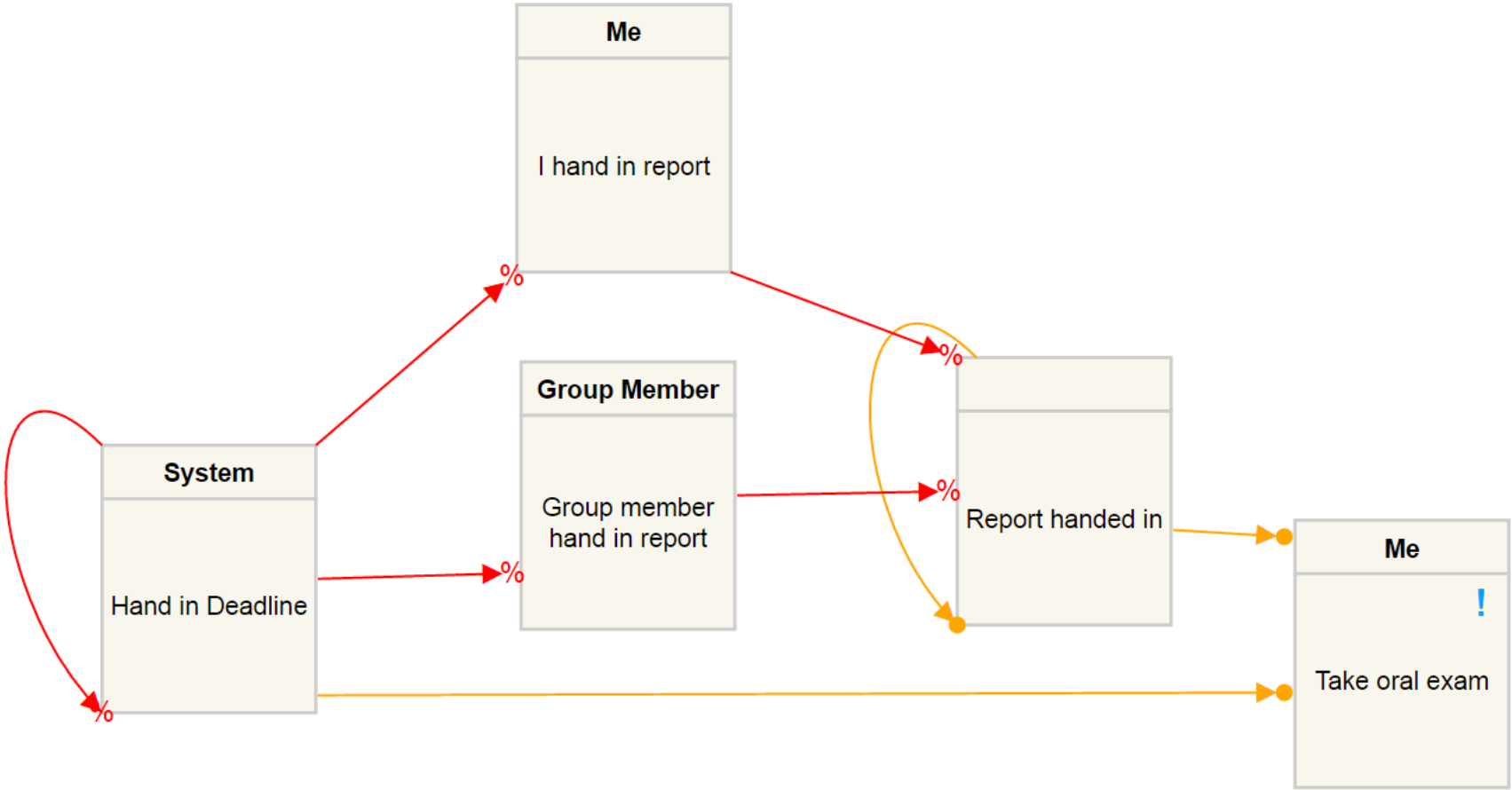

BAD: Lifelocked, the pools sends messages to each other that forces a continuation of the loops in each pool.


C
A
B
D

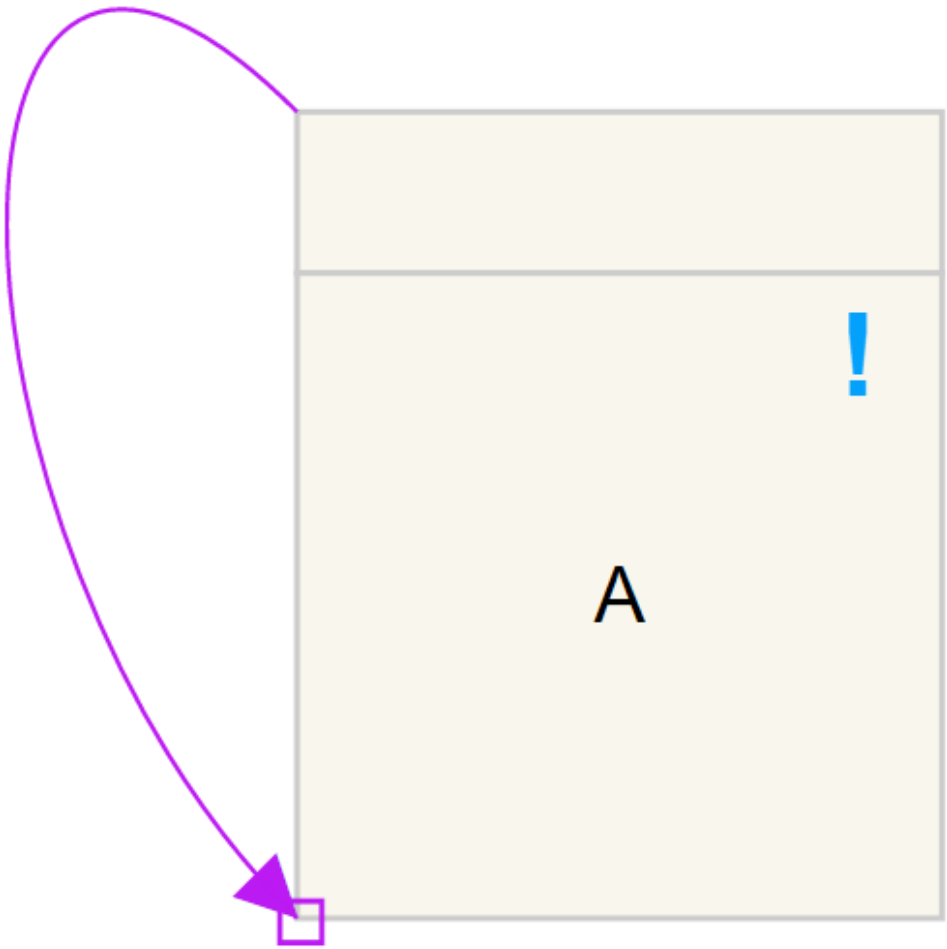


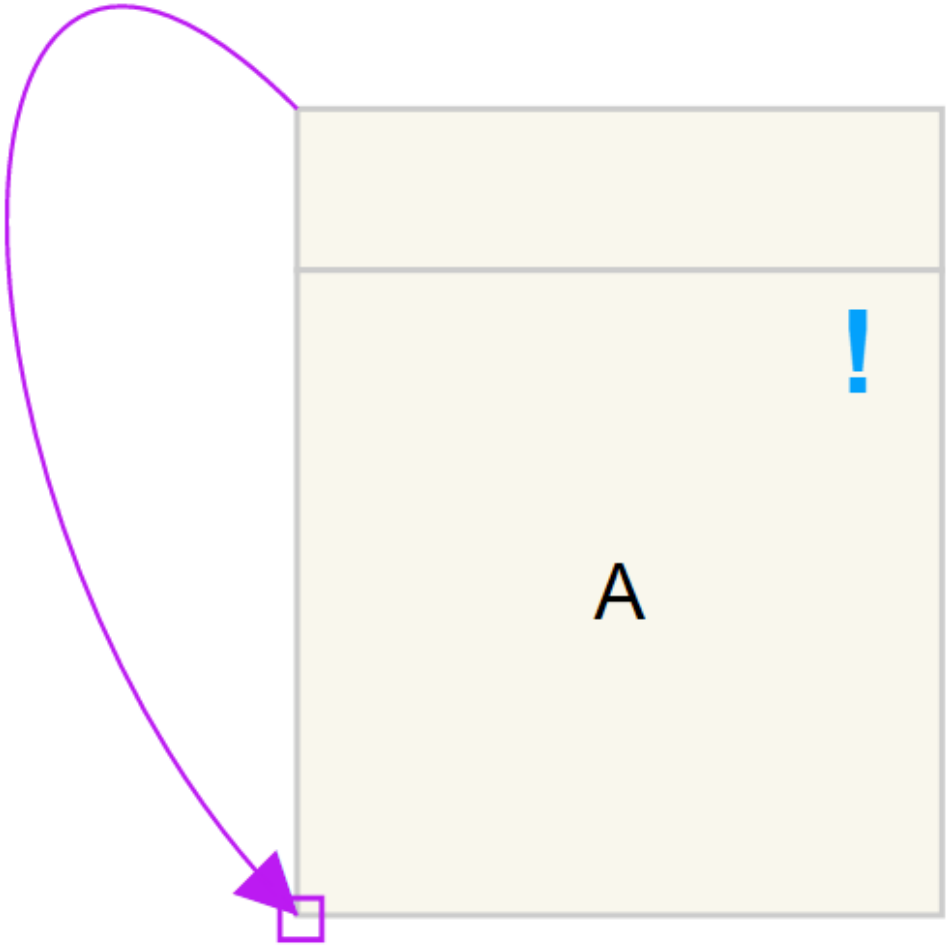

Possible BAD: livelocked, if A happens before C then the graph gets deadlocked, else it is fine.


Me
I hand in report
Group Member
Group member hand in report
System
Hand in Deadline
Report handed in
Me
!
Take oral exam




Not sound but Good: dead activity, The activity "report handed in" is not able to be executed ever, but it is by design so first to hand in allows "me" to go up to the oral exam.


!
A




BAD: Deadlocked and dead activity, the milestone blocks A from being able to execute due to it being pending.

# Unstructured vs Structured processes

- A well-structured or block-structured process model is composed of blocks (Single-Entry Single-Exit fragments), which can be nested, but must not overlap

# Unstructured vs Structured processes

- A well-structured or block-structured process model is composed of blocks (Single-Entry Single-Exit fragments), which can be nested, but must not overlap
- Blocks can be single activities, sequences, parallel branchings, alternative branchings, loop blocks or the entire process model

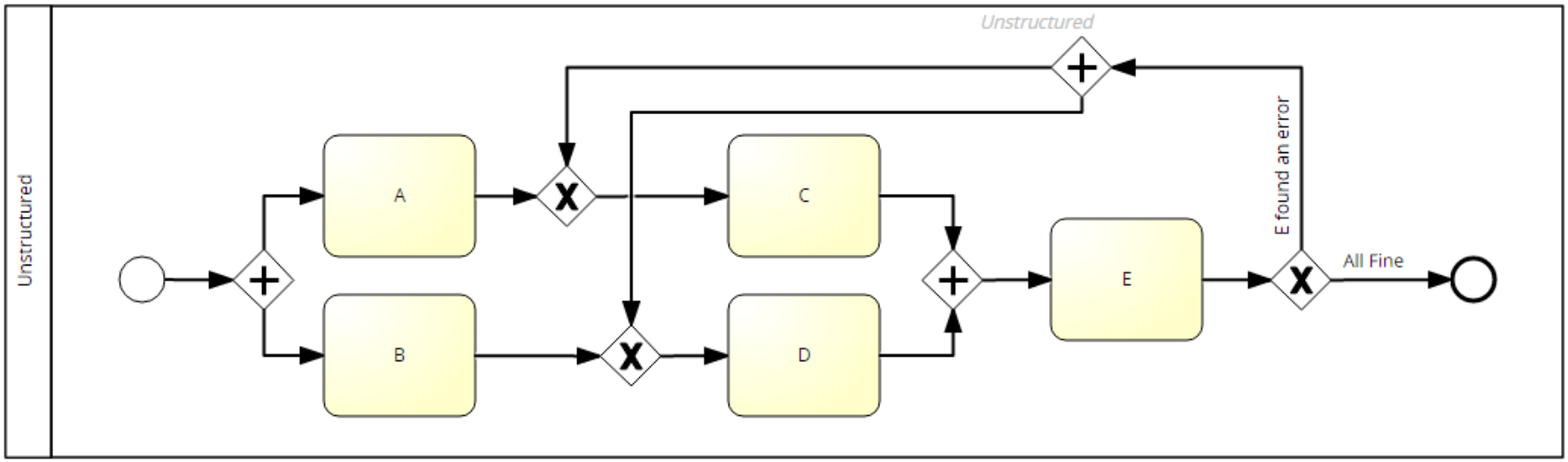
# Unstructured vs Structured processes

- A well-structured or block-structured process model is composed of blocks (Single-Entry Single-Exit fragments), which can be nested, but must not overlap
- Blocks can be single activities, sequences, parallel branchings, alternative branchings, loop blocks or the entire process model
- Unstructured models require advanced verification techniques

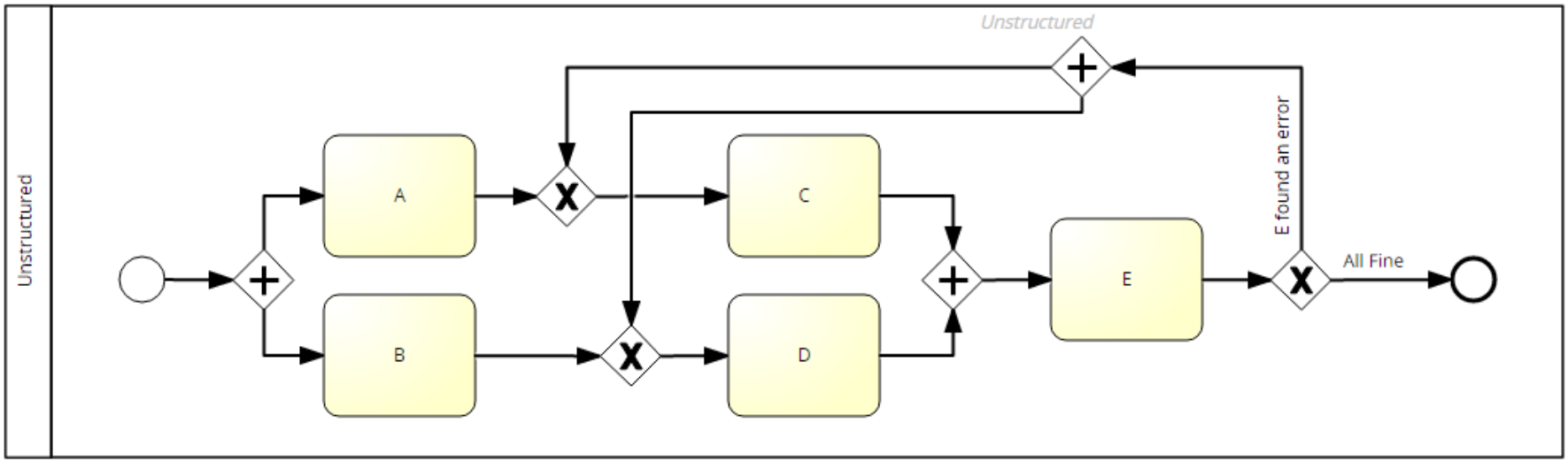
# Unstructured vs Structured processes

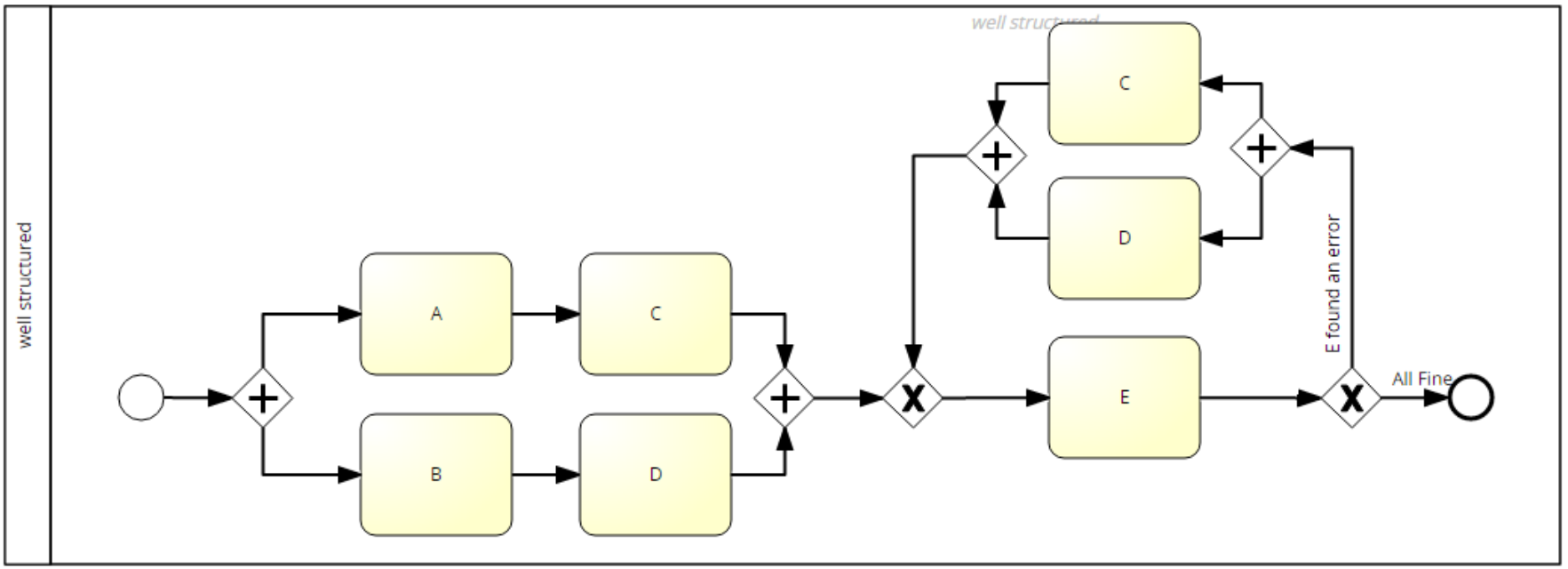
- A well-structured or block-structured process model is composed of blocks (Single-Entry Single-Exit fragments), which can be nested, but must not overlap
- Blocks can be single activities, sequences, parallel branchings, alternative branchings, loop blocks or the entire process model
- Unstructured models require advanced verification techniques
- Unstructured models are more difficult to understand and are a considerable source of errors

# Unstructured vs Structured processes



# Unstructured vs Structured processes



# Transition Systems

- A Transition system, is a model where one model transitions between states of a process.

# Transition Systems

- A Transition system, is a model where one model transitions between states of a process.
- A state is a snapshot of the process including which values each data object may have, and what state each activity/event have.

# Transition Systems

- A Transition system, is a model where one model transitions between states of a process.
- A state is a snapshot of the process including which values each data object may have, and what state each activity/event have.
- States can also be marked as initial (the state the process starts in) or accepting (a state which allows the process to end).

# Transition Systems

- A Transition system, is a model where one model transitions between states of a process.
- A state is a snapshot of the process including which values each data object may have, and what state each activity/event have.
- States can also be marked as initial (the state the process starts in) or accepting (a state which allows the process to end).
- Transitions are when activities in the process is executed. The execution of an activity may cause a transition to a new state or to the same state.

# State space explosion

- As a state is a snapshot of a process, does it means that all possible snapshots are possible states.

# State space explosion

- As a state is a snapshot of a process, does it means that all possible snapshots are possible states.
- For a dcr-graph each event may have 8 different states. This makes it so that for a full graph we MAY have up to $8^n$ states, where $n$ is the number of events in the graph.

# State space explosion

- As a state is a snapshot of a process, does it means that all possible snapshots are possible states.
- For a dcr-graph each event may have 8 different states. This makes it so that for a full graph we MAY have up to $8^n$ states, where $n$ is the number of events in the graph.
- In a transition system not all states are reachable.

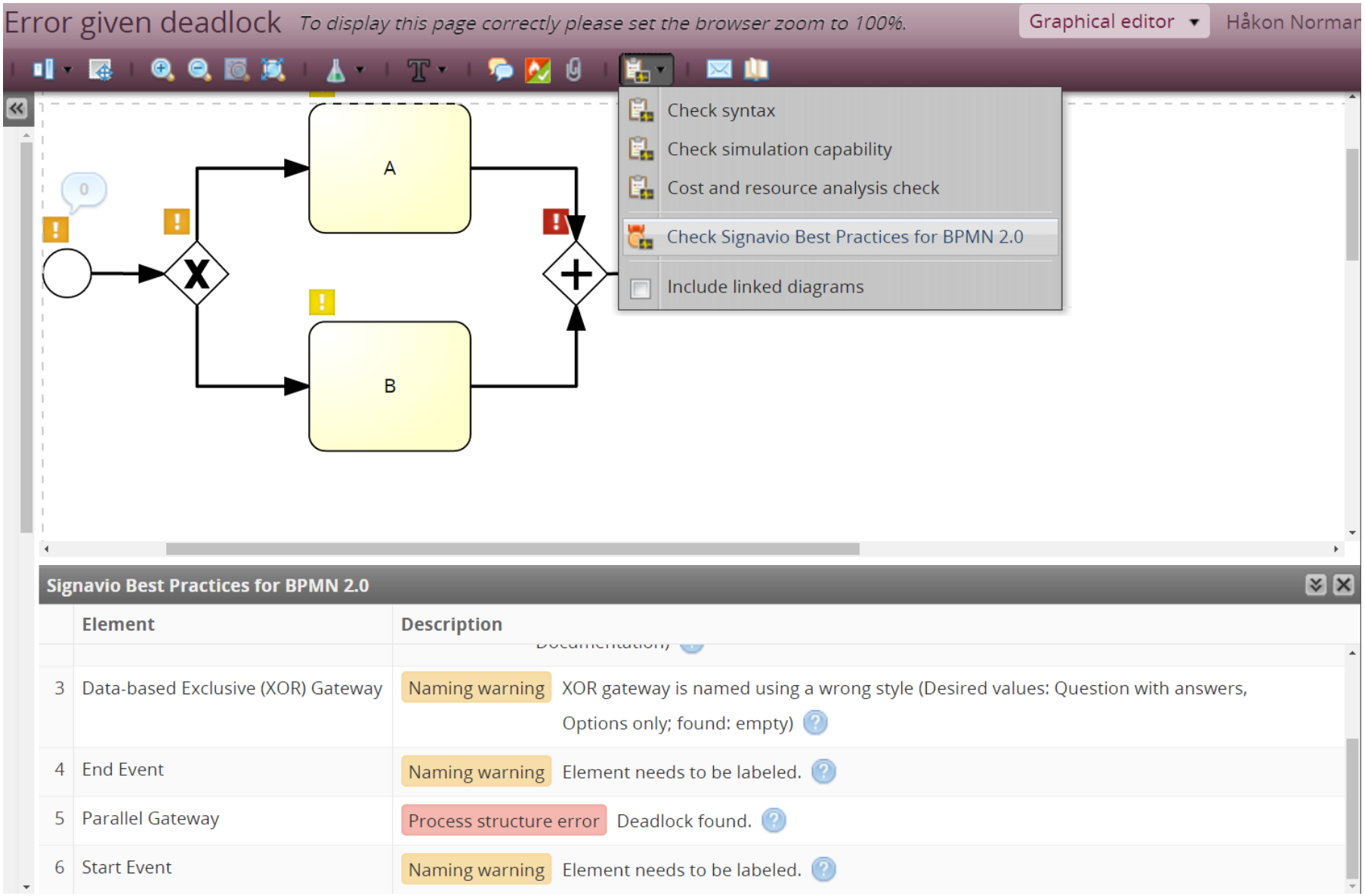
# Checking for deadlocks and livelocks

- Deadlocks
  - From an initial state do a breadth-first search over states reachable by transitions from the initial state. to look for states with no outgoing transition

# Checking for deadlocks and livelocks

- Deadlocks
  - From an initial state do a breadth-first search over states reachable by transitions from the initial state. to look for states with no outgoing transition
- Livelocks
  - Do a breadth-first search to look for loop backs to earlier visited states.

# Signavio checks



# Signavio checks