

BPMN Meets DMN: Business Process and Decision Modeling

*Week 3:
Analyzing Process Behavior*

An openHPI Course

Prof. Dr. Mathias Weske

Hasso Plattner Institute at the University of Potsdam
Business Process Technology

Week 3

Analyzing Process Behavior

- 3.1 Process Behavior
- 3.2 Structural Soundness
- 3.3 Soundness Explained
- 3.4 Simulating Business Processes
- 3.5 Informal Petri Net Primer (Bonus video)
- 3.6 Mapping BPMN to Petri Nets (Bonus video)
- 3.7 Checking Soundness (Bonus video)

In the bonus homeworks, you can own additional points

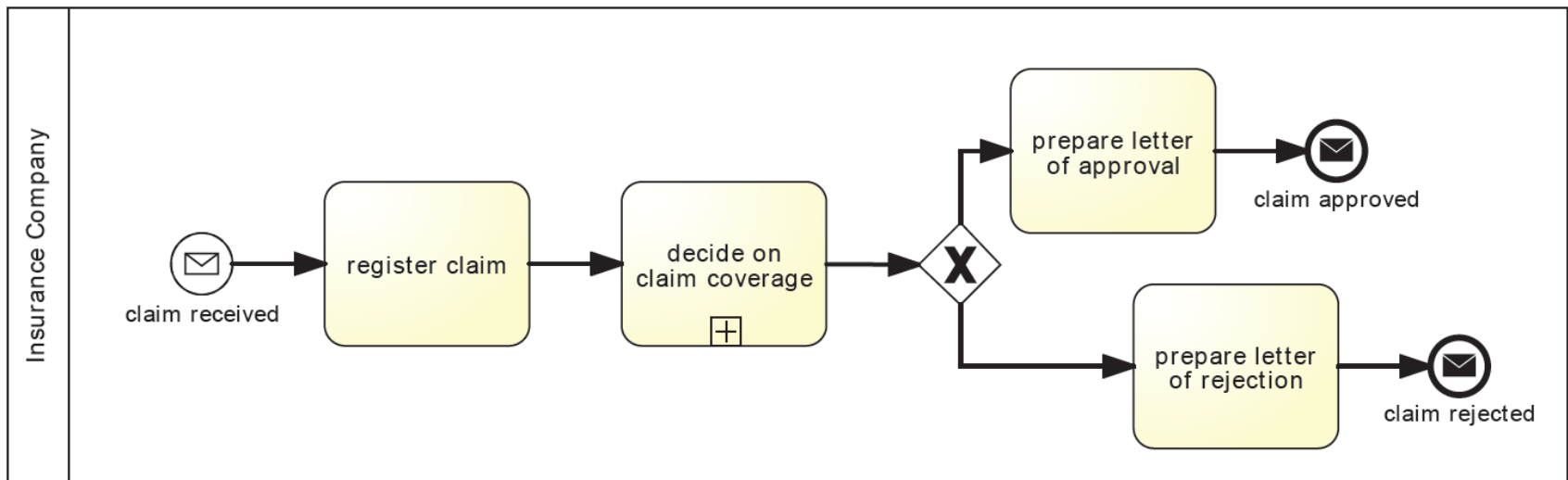
Video Clip 3.1

Process Behavior

BPMN Meets DMN: Business Process and Decision Modeling:
An openHPI Course by Mathias Weske

Foundations of Modeling Languages

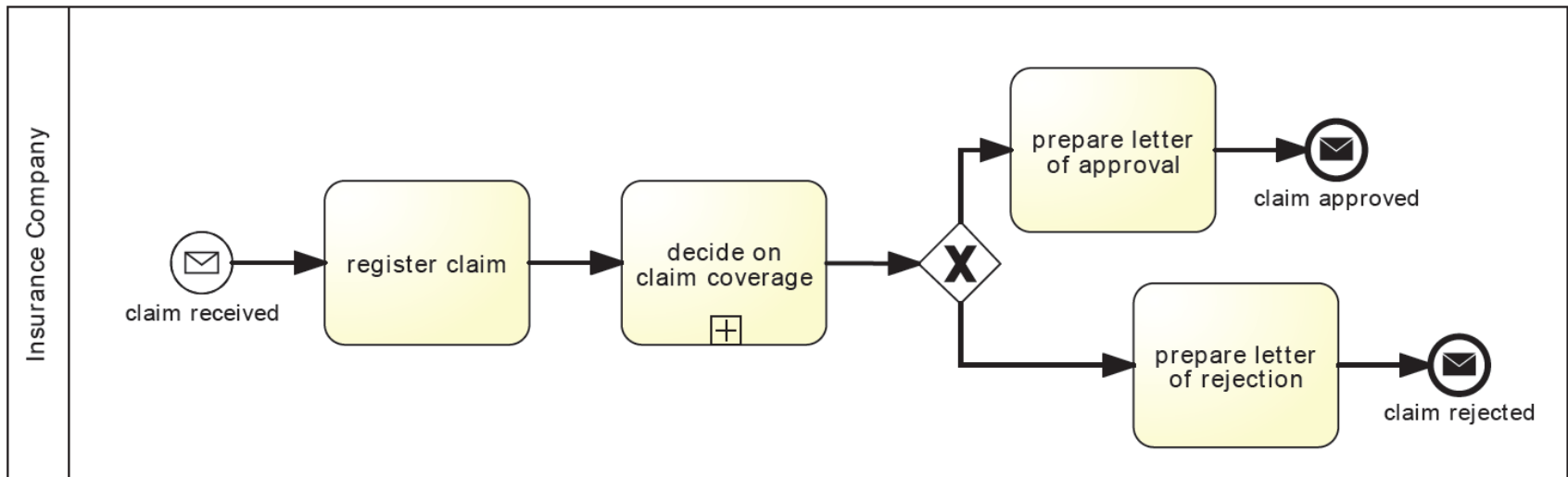
- Week 2: Syntax and semantics of modeling languages



- Syntax
 - Syntactic correctness makes sure that the elements of the model are organized in a way that matches the abstract syntax of the modeling language.

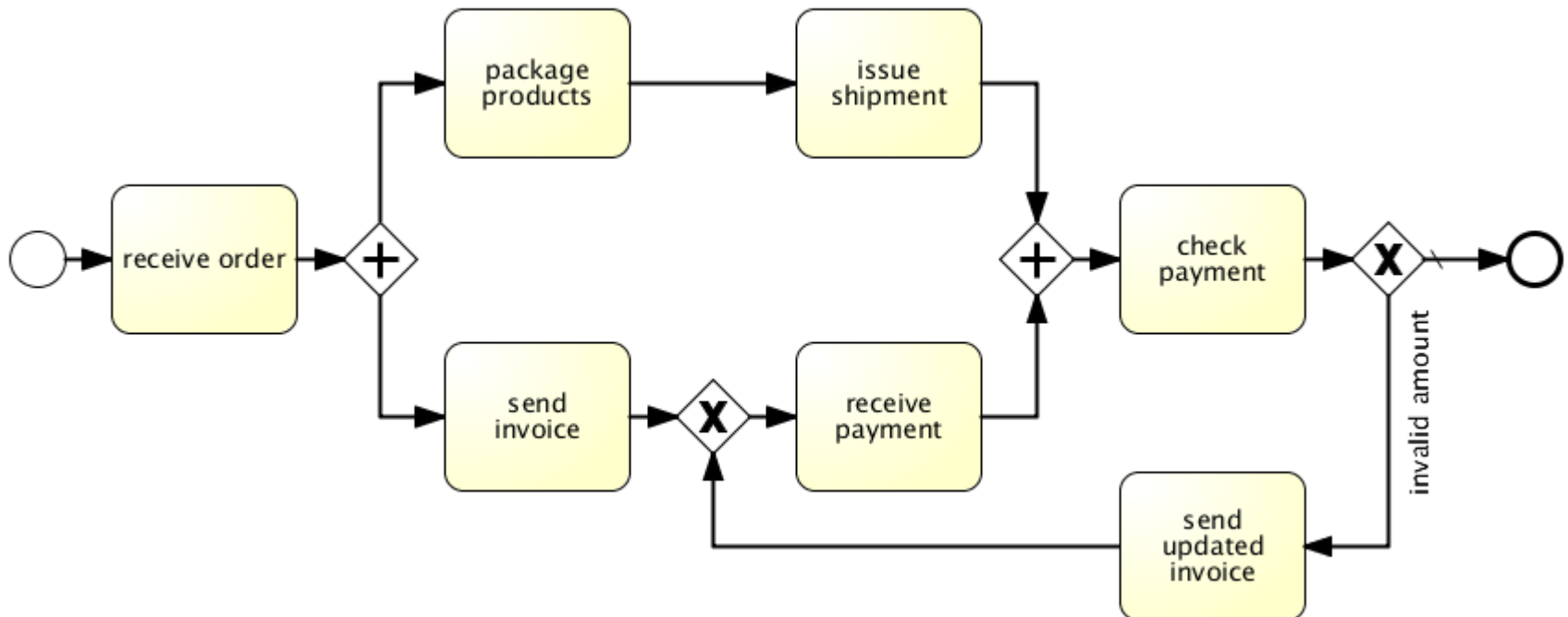
Semantics of Process Models

- Semantics
 - Defines the meaning of syntactically correct process models
 - The meaning is the behavior defined by the process model, i.e., the order of performed activities



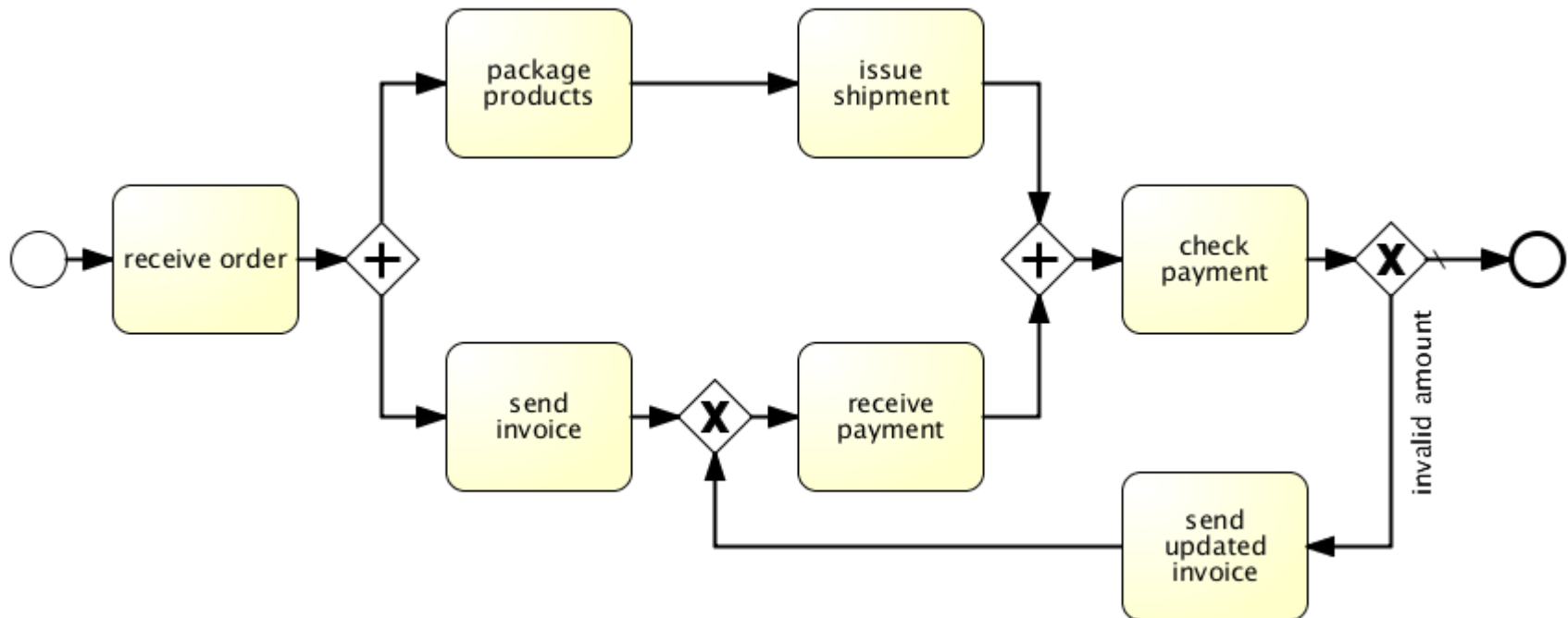
Beyond Syntactical Correctness

- Semantic correctness
 - Ensures that a process behaves correctly and terminates properly
 - To analyze semantic correctness, the process model needs to be checked for the absence of behavioral anomalies



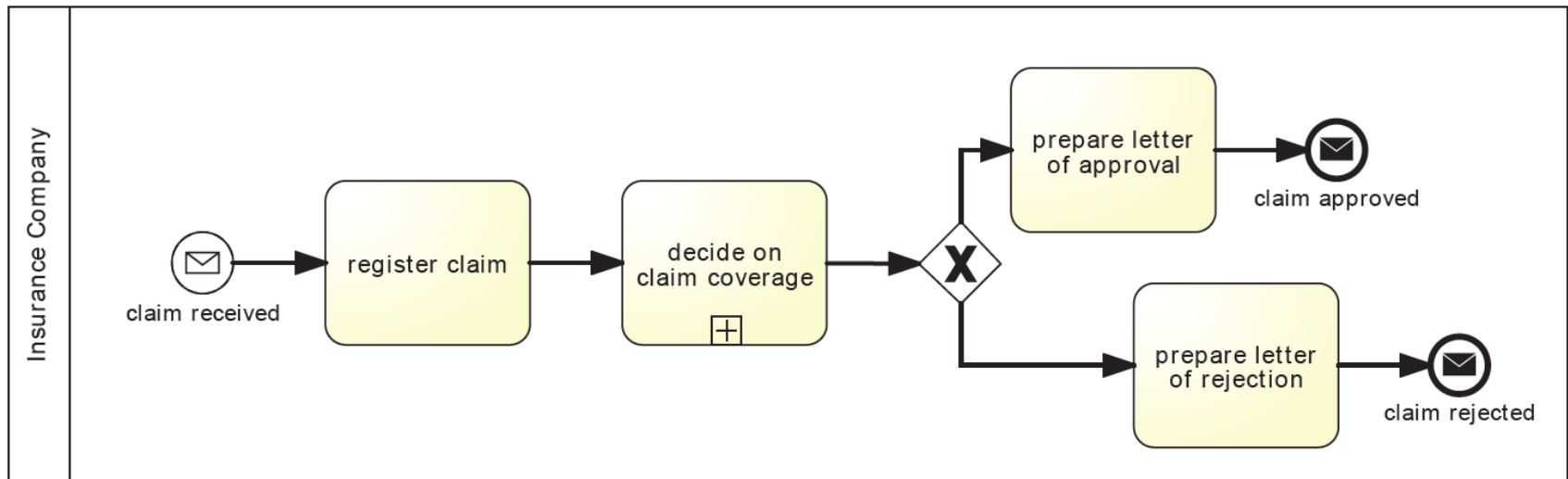
Beyond Syntactical Correctness

- In this week
 - We explain correct process behavior informally
 - As bonus material, we also introduce Petri nets and a mapping from BPMN to Petri nets for a formal investigation of correctness

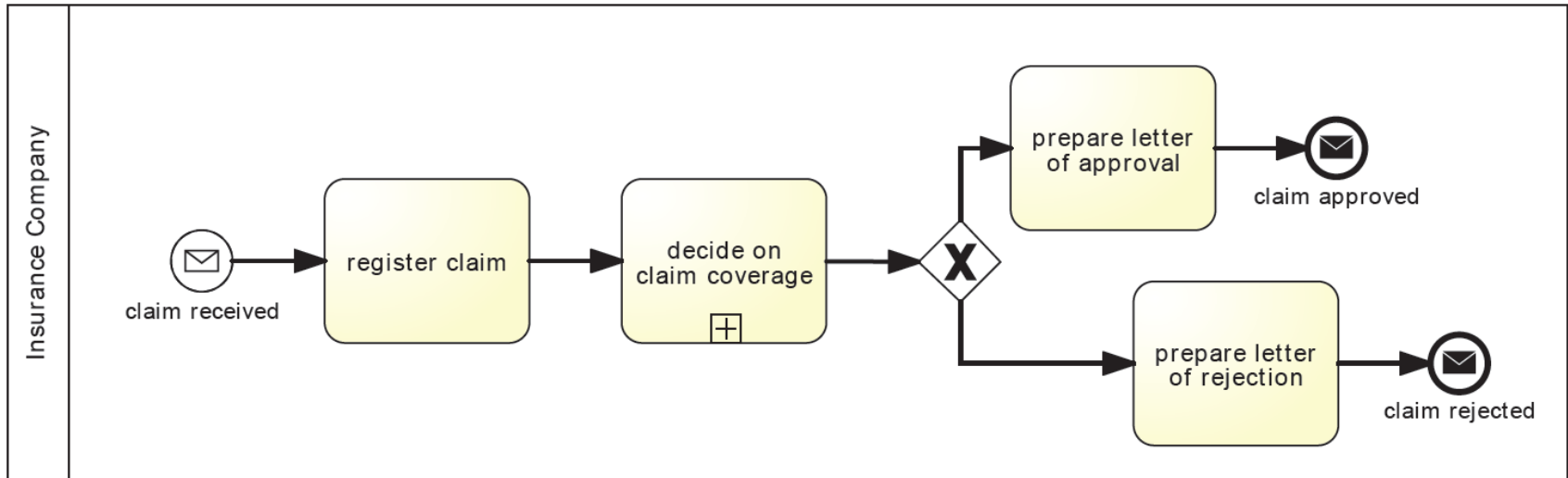


Process Behavior

- The behavior of a process model can be characterized by the set of all execution sequences that are possible
 - We look at the traces of the process model, resulting from events that occur and activities that are performed
 - The resulting semantics is called *trace semantics*



Process Behavior



- Example traces
 - claim received, register claim, decide on claim coverage, prepare letter of approval, claim approved
 - claim received, register claim, decide on claim coverage, prepare letter of rejection, claim rejected

Process Behavior

- In behavioral analysis, we abstract from the actual running state of an activity instance
 - This is appropriate, since we are interested in the logical ordering of activities, not in the activities themselves

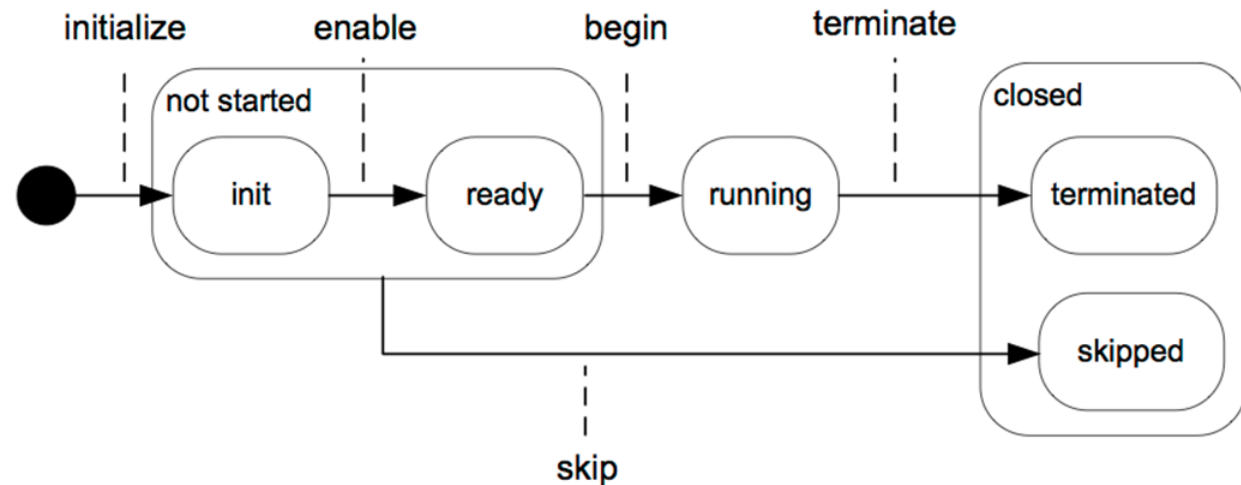
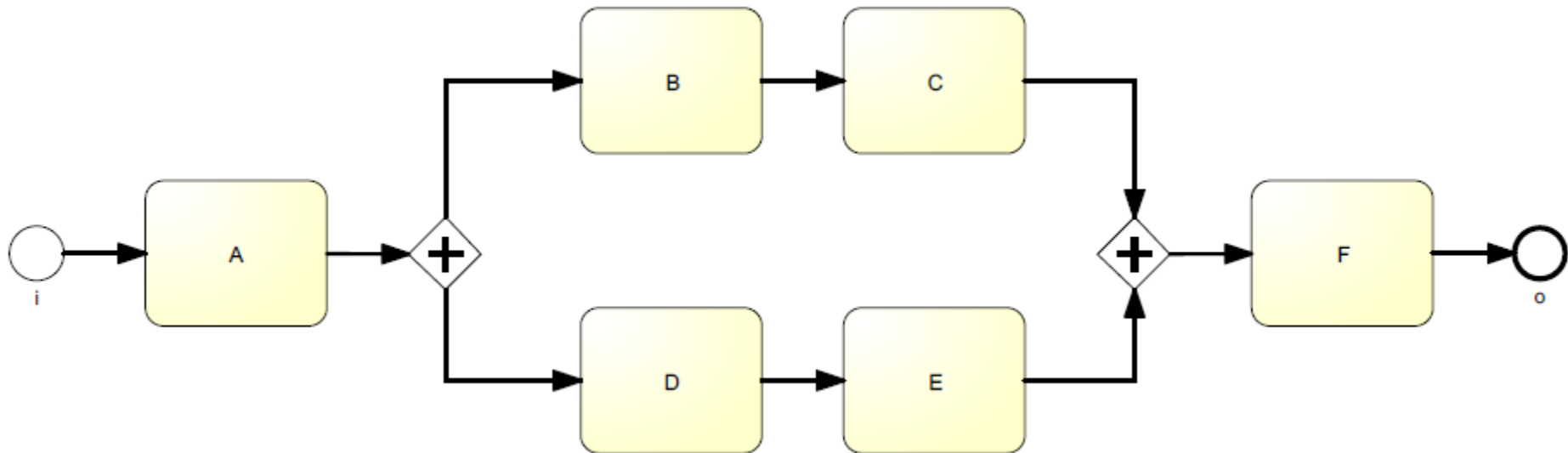


Fig. 3.9. State transition diagram for activity instances

Process Behavior

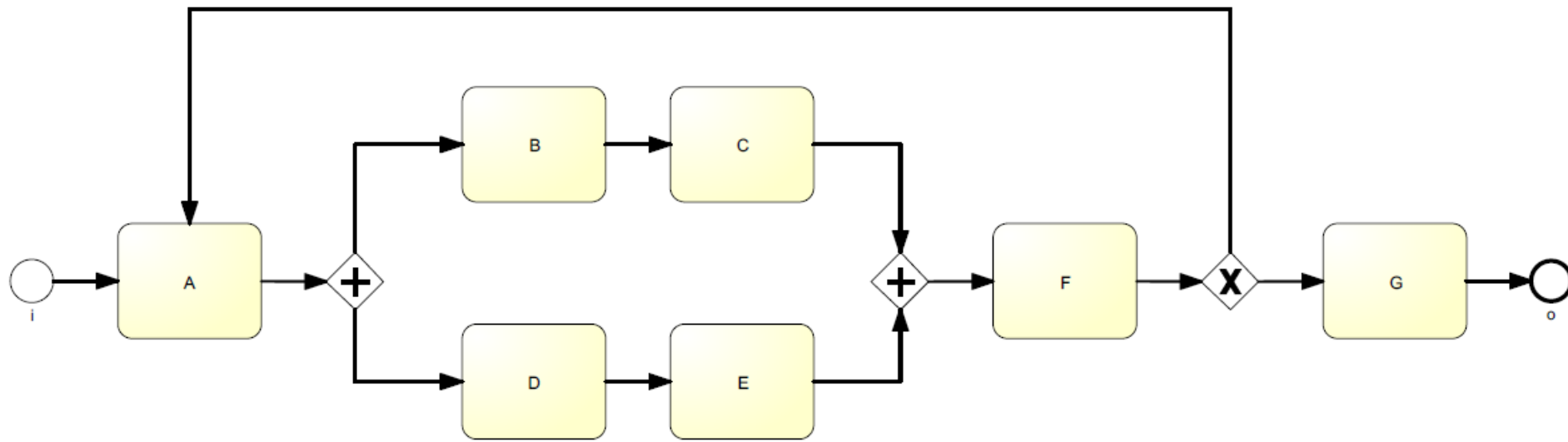
- Concurrency results in traces with an arbitrary ordering of concurrent activities
- A more abstract example
 - iABDCEFo, iABCDEFo, iADEBCFo, ...



- In general
 - iA followed by any ordering of B,C,D,E, where B occurs before C and D occurs before E, followed by Fo

Process Behavior

- Loops result in traces of arbitrary length
- Example
 - One loop iteration: iABDCEFADEBCFGGo, ...
 - Two loop iterations: iABDCEFADEBCFADEBCFGGo, ...
 - Any number of loop iterations possible



Video Clip 3.1

Process Behavior

- Syntactical correctness
- Semantic correctness
- Behavioral anomalies
- Traces as a means to characterize behavior

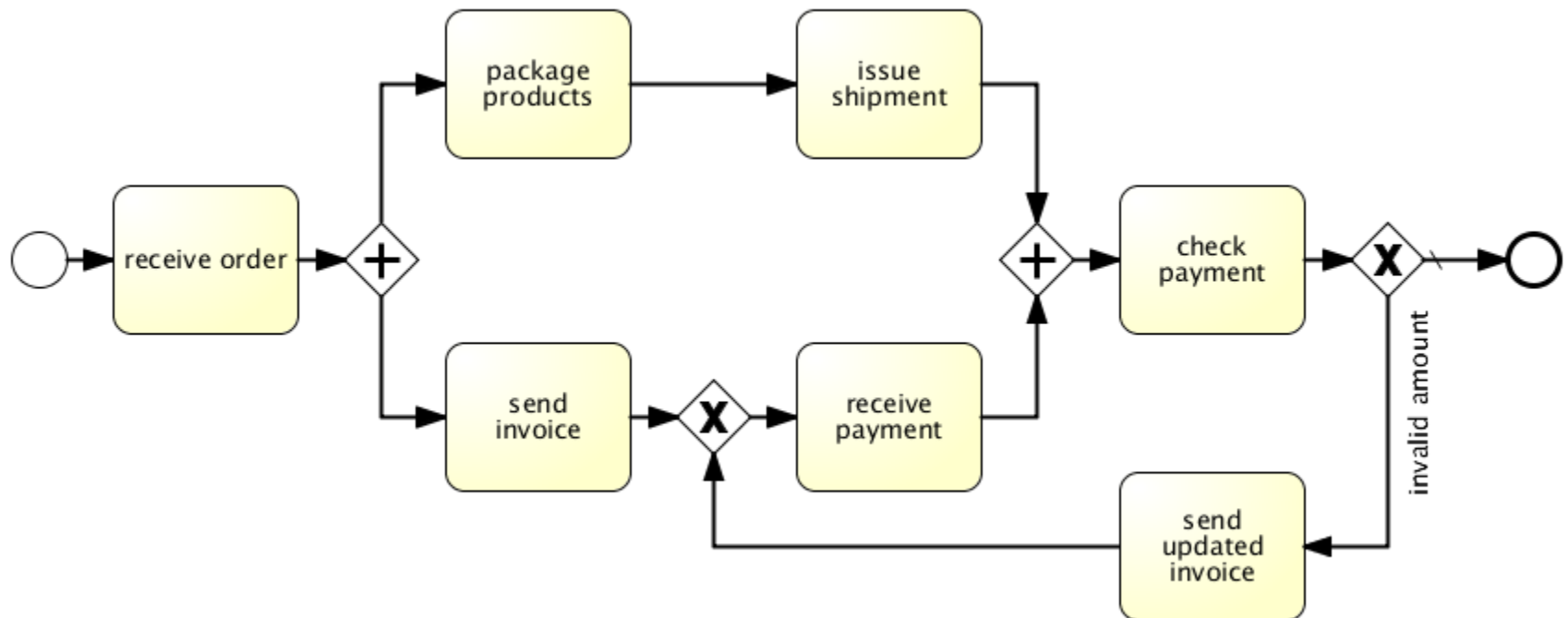
Video Clip 3.2

Structural Soundness

BPMN Meets DMN: Business Process and Decision Modeling:
An openHPI Course by Mathias Weske

Soundness Criterion

- Soundness is a generic correctness criterion for process models, formally specified using Petri nets
 - Developed by Wil van der Aalst
 - In this course, we transfer soundness to the level of BPMN

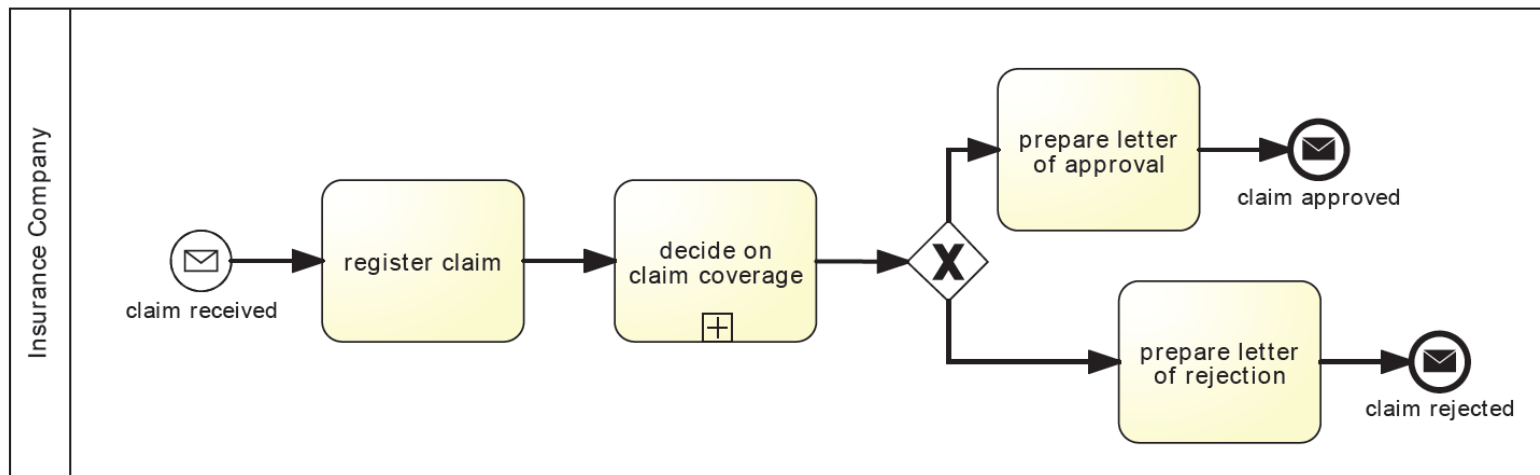
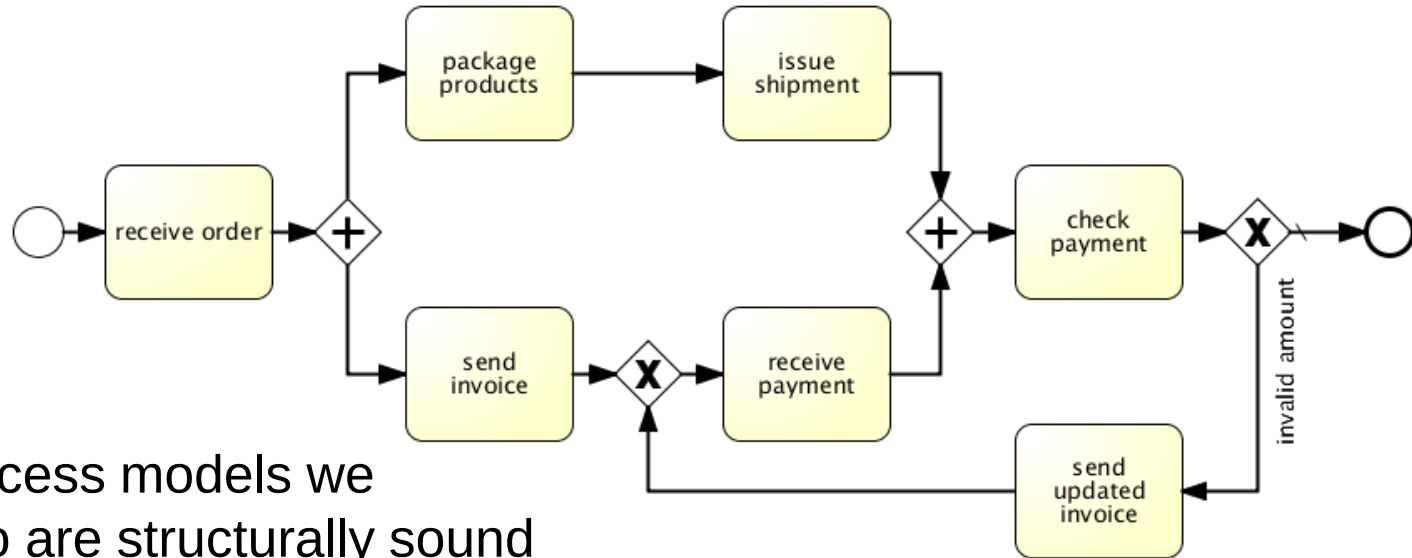


Structural Soundness

- Soundness assumes that a process is structurally sound
- A process model is structurally sound, if
 - it has exactly one start event i (for „input“ of the process),
 - it has exactly one end event o (for process „output“), and
 - each node in the process model is on a path from i to o
- Structural soundness is a necessary condition for soundness, but not a sufficient condition
 - If a process model is not structurally sound, it cannot be sound

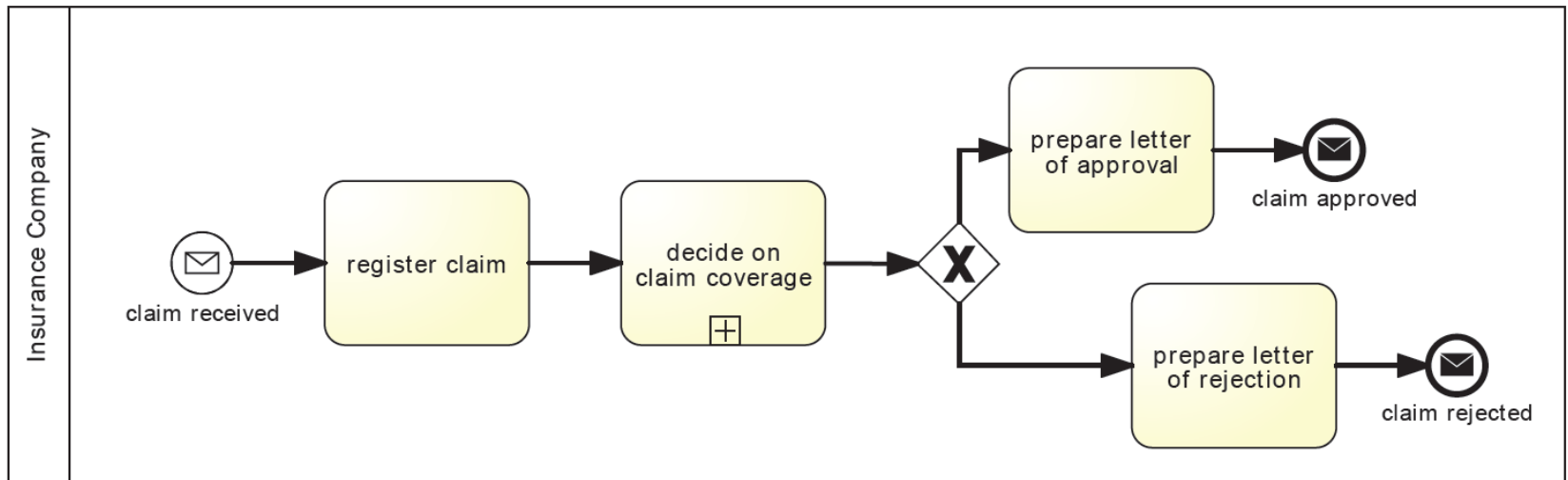
Structural Soundness

- Notice
 - Several process models we looked at so are structurally sound
 - Some are not



On Structural Soundness

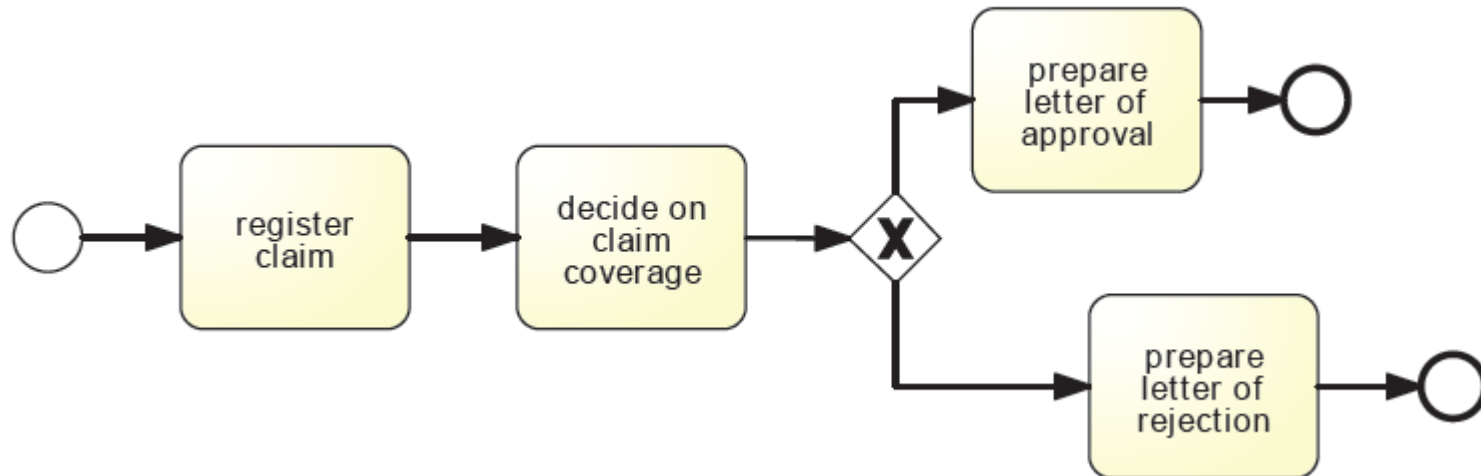
- Process models with multiple end events are not structurally sound



- All these process models cannot be not sound
 - Even if they do not exhibit any behavioral anomalies
 - Therefore, we transform these models to structurally sound process models without changing their behavior

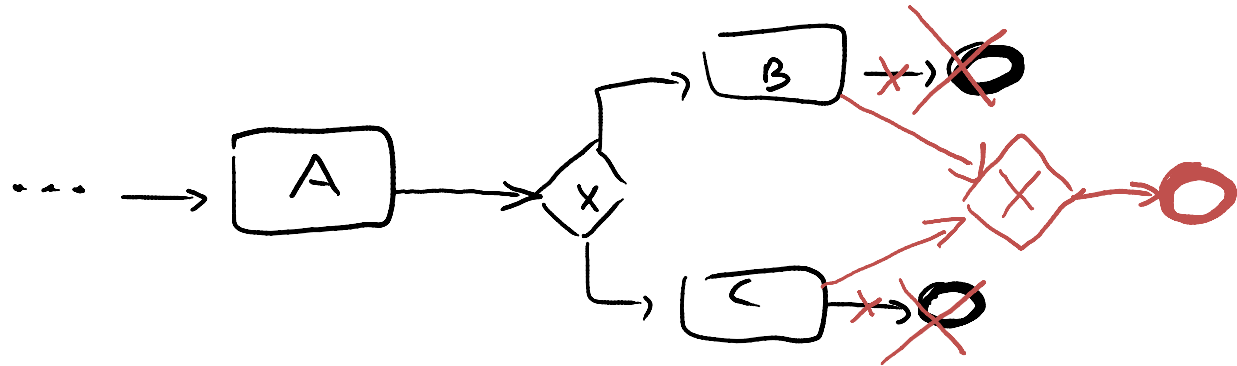
On Structural Soundness

- To start formal analysis, we
 - delete the labels, event information, and the role(s) involved



- Now we transform structurally unsound process models to structurally sound process models
- Assumption
 - All end events of the process model are exclusive, i.e., there is no trace in which more than one end event occurs
 - If this assumption is not met, the transformation introduces errors

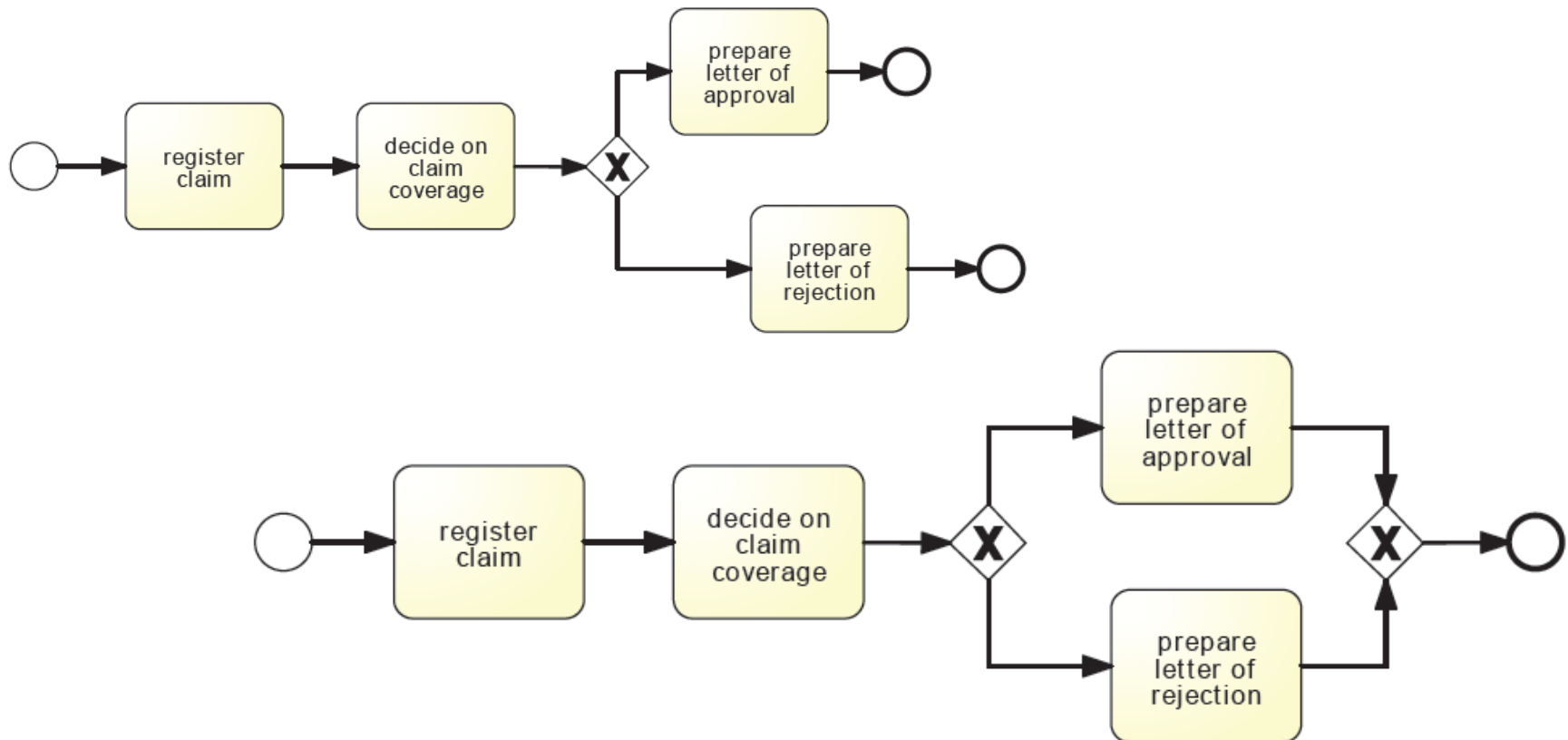
On Structural Soundness



- Process models with exclusive end events can be translated to structurally sound process models by
 - deleting the end events,
 - introducing an exclusive gateway that joins the end event branches, and
 - introducing a new end event.

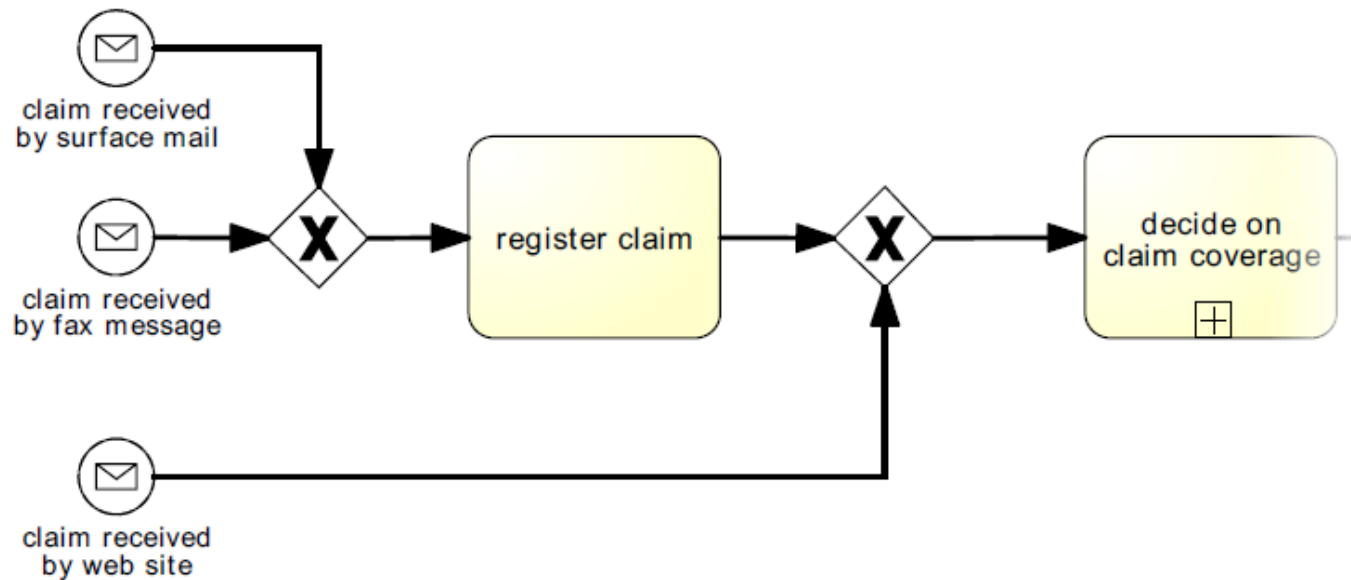
On Structural Soundness

- Notice
 - The different outcomes of the process are no longer represented explicitly in the process model
 - Since we use the transformed model only for analysis, this is fine



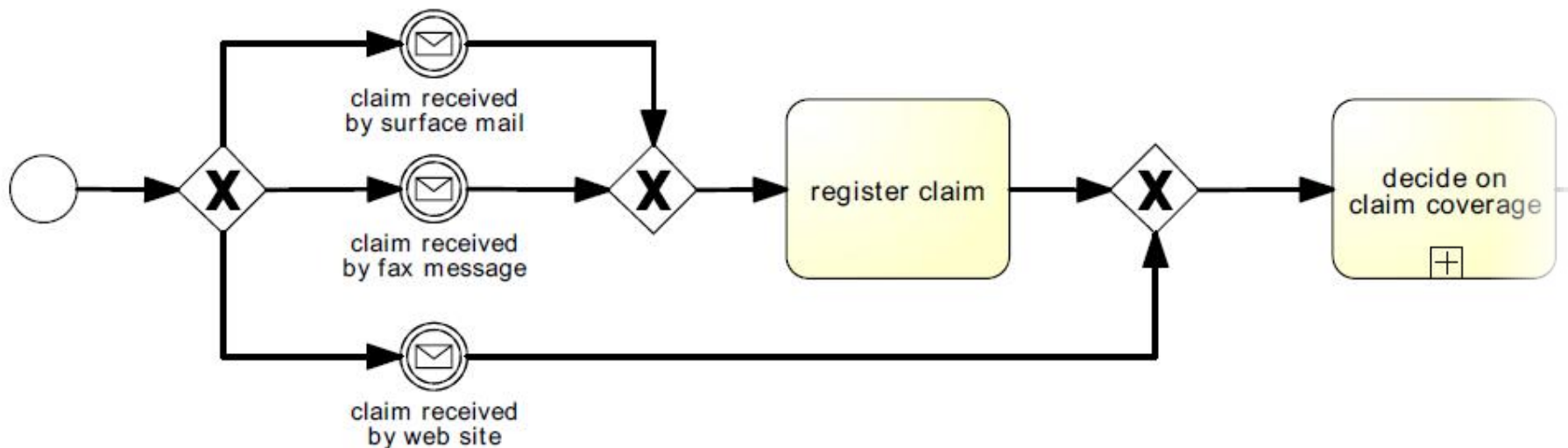
Multiple Start Events

- In practice, we also see process models with multiple start events



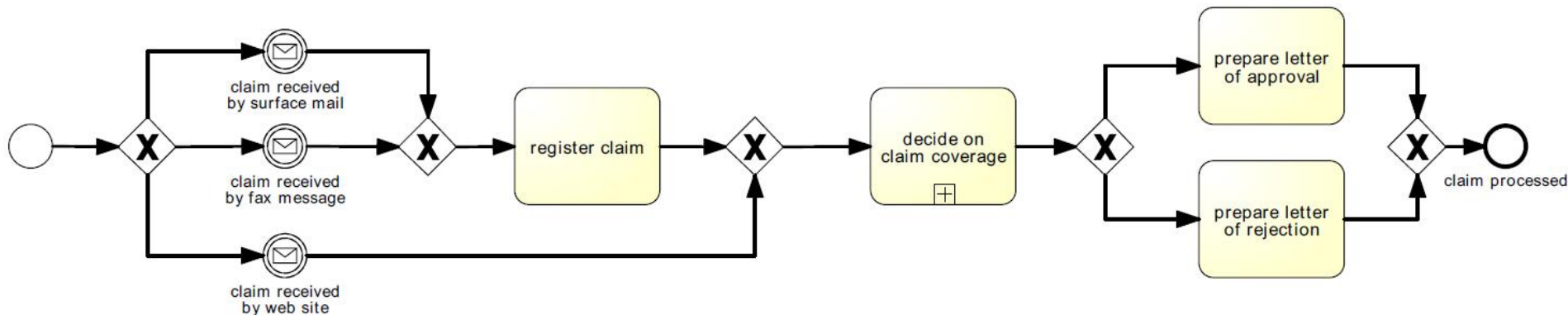
How to transform

- Also in this case, we can transform processes with alternative start events to be structurally sound
 - Add a new start event and an exclusive gateway
 - Connect it to all alternative start events, which become thereby intermediate events



How to transform

- These transformation operations lead to a process model that is structurally sound



- Notice
 - This transformation is only for the purpose of checking soundness
 - The process model is not very useful, mainly due to the „magic“ decision in the beginning
 - However, it is useful for soundness checking

Video Clip 3.2

Structural Soundness

- Behavioral analysis requires structural soundness
 - Single start event i
 - Single end event o
 - Each node is on a path from i to o
- Formal analysis abstracts from labels and roles
- Process models with multiple start and multiple end events can be transformed, if events are exclusive

Video Clip 3.3

Soundness Explained

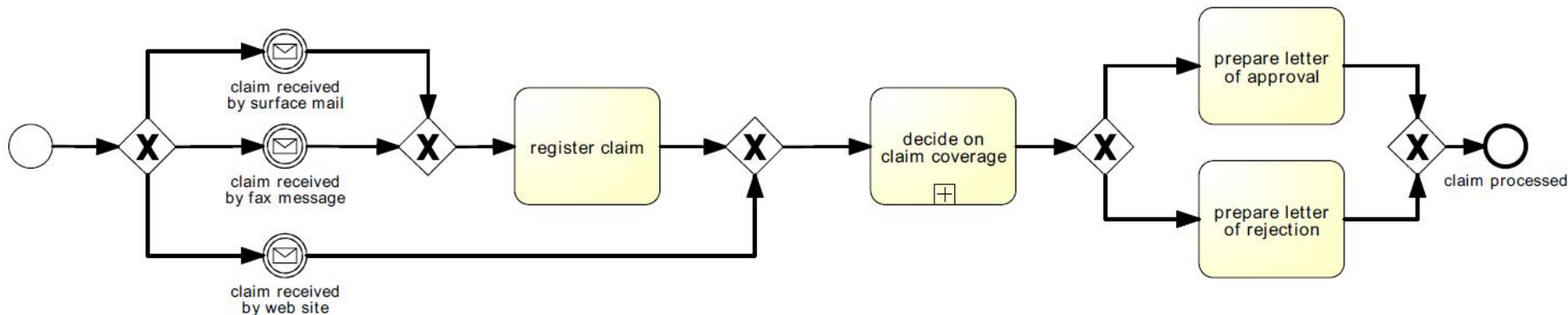
BPMN Meets DMN: Business Process and Decision Modeling:
An openHPI Course by Mathias Weske

Idea of Soundness

- We assume process models to be structurally sound
- Soundness
 - (1) Once a process has started, regardless of which decisions are taken by the process, at some point, the process will reach the end event o
 - (2) If and when it reaches o , the process has completely terminated, i.e., no further activities can be executed by the process anymore
 - (3) Each process activity participates in at least one execution (in one trace)
- Notice
 - Soundness is a generic correctness criterion, similar to normal forms in databases
 - It makes sure that all process instances behave correctly

Soundness Properties

- Properties (1), (2), and (3) by example



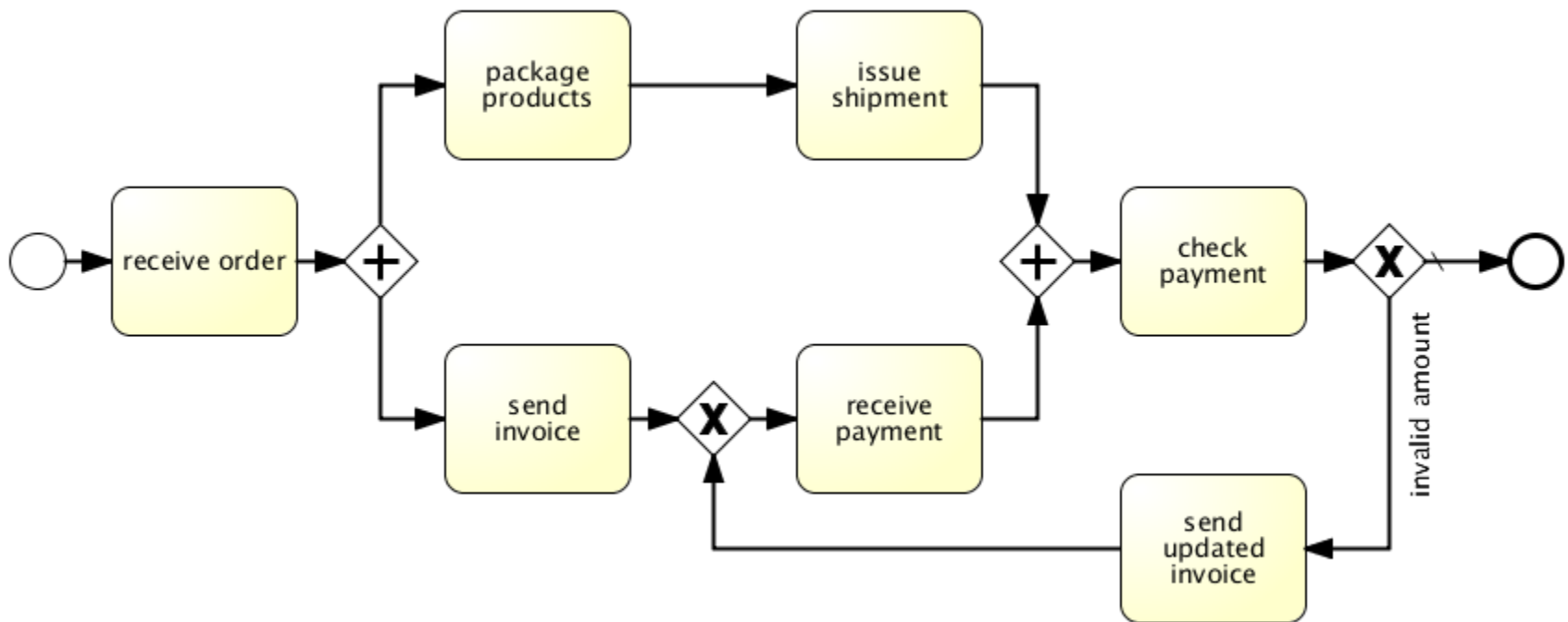
- Formally, the process is in a deadlock after o has been reached
 - But this behavior is intended, so this does not violate soundness!

Unsound Behavior

- We can detect unsound process behavior by finding a violation of either of the soundness properties
- Violation of (1)
 - The process cannot reach the final event o, it gets stuck before
 - Deadlock (before o is reached)
- Violation of (2)
 - Remaining activity in the process after reaching o
 - Or even infinite activity, livelock
- Violation of (3)
 - Activities that can never be executed
 - Dead activities

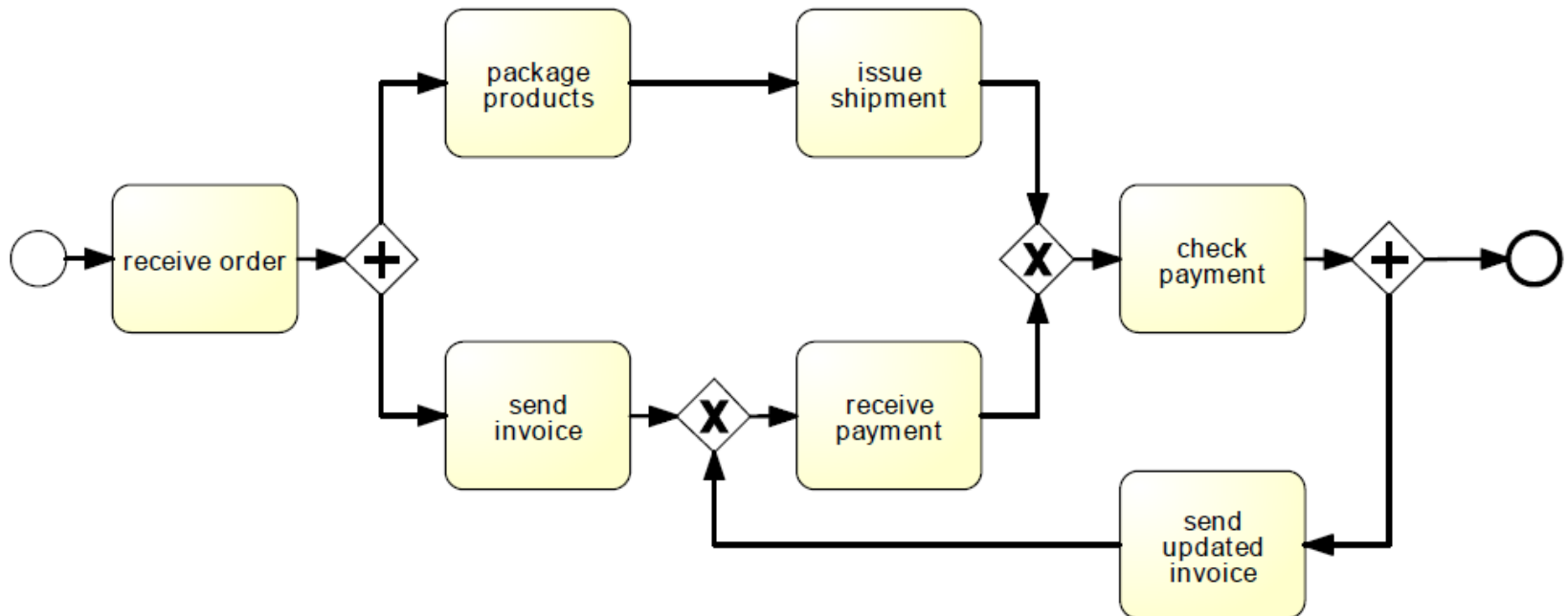
Violation of (1): Deadlock

- Since the „*invalid amount*“ decision can be taken, the process model can deadlock before o is reached
 - This process model violates property (1)



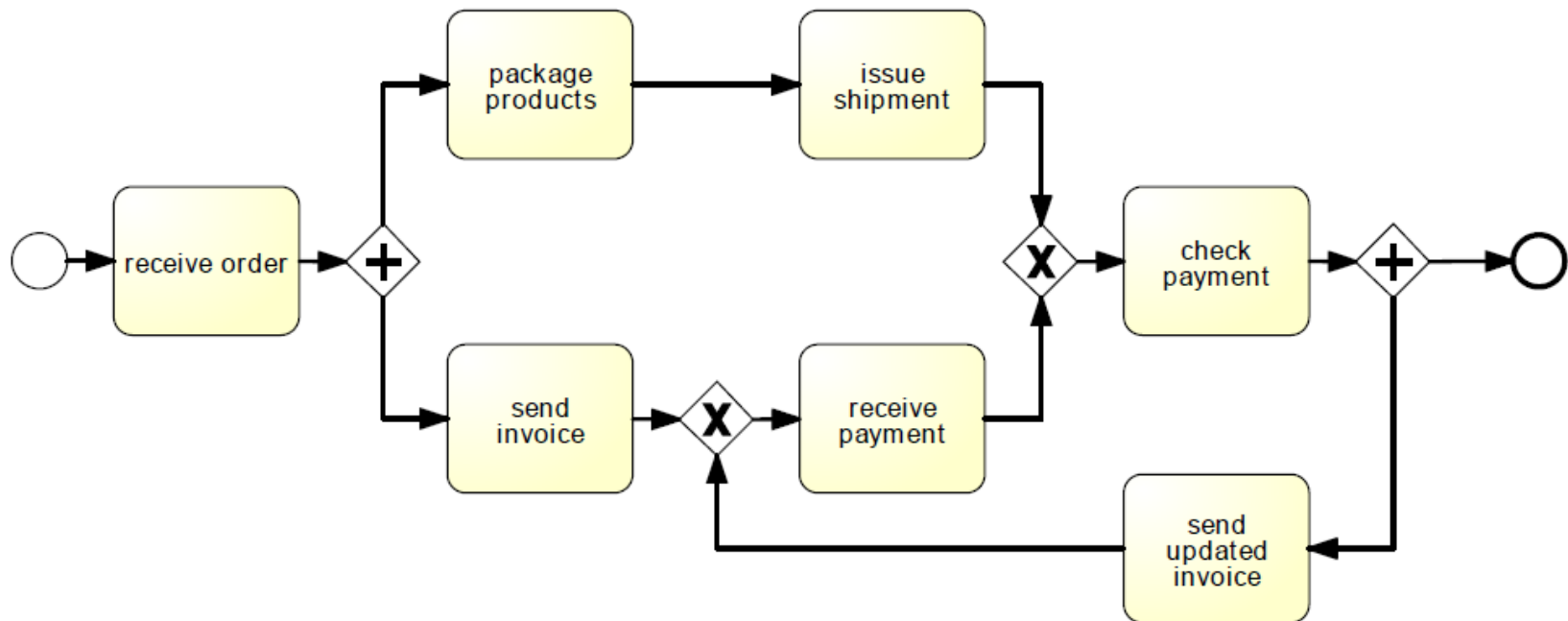
Violation of (2): Livelock

- (2) If and when the process reaches o, it has completely terminated, i.e., no further activities can be executed by the process anymore
- The process continues infinitely, even after reaching o



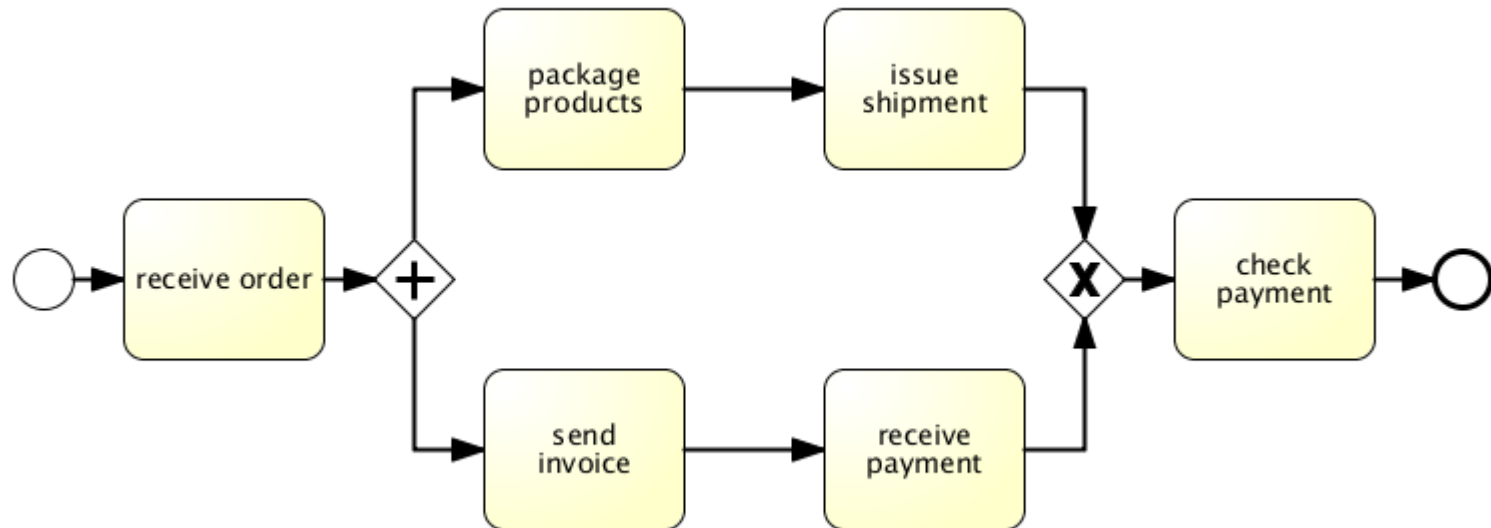
Livelock

- A process is in a livelock, if the control flow structure makes the process executing an infinite number of activities



Violation of (2): Remaining Activity

- A process suffers from remaining activities, if process activities can be executed after o has been reached



- Please notice
 - Validation of (3) is rather seldom in real world process models, so we do not cover it

How to avoid unsound processes

- Always use structured process models
 - Each split gateway has exactly one join gateway of the same type so that each path originating from the split will reach the join
 - Makes also sure that loops obey the block structure, see earlier example
- However
 - This constraint limits the possibilities of modeling
 - Process models that are best understood might not be possible
- Better way
 - Use smart modeling tools that provide soundness checks and feedback messages on why a given process is not sound

Video Clip 3.3

Soundness Explained

- Soundness
 - Every process instance terminates properly
 - Each process activity can participate
- Violations
 - Deadlocks
 - Livelocks
 - Remaining activities
- Structured process models help avoid these anomalies

Video Clip 3.4

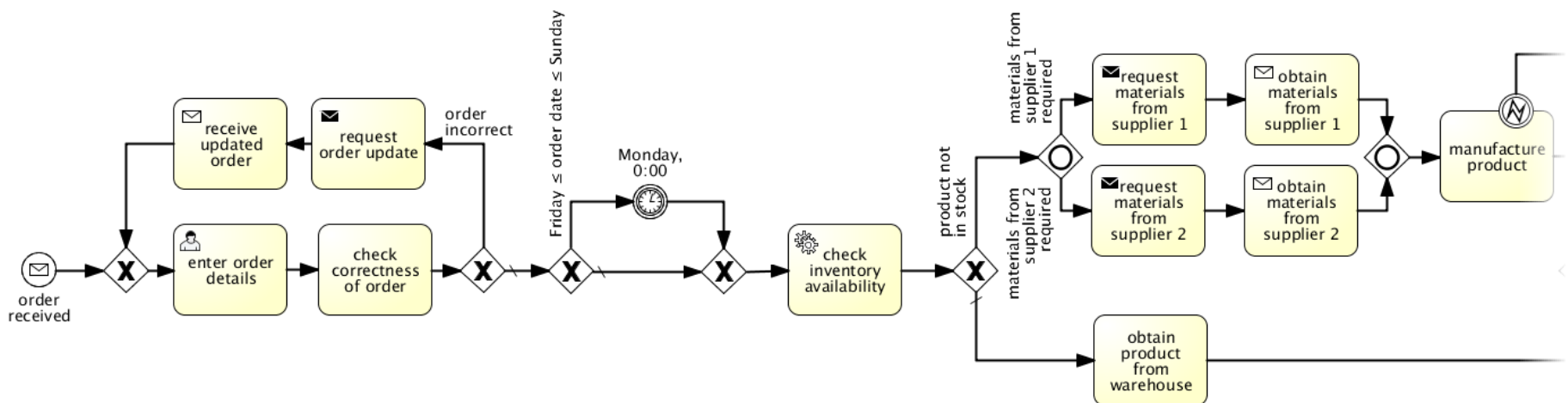
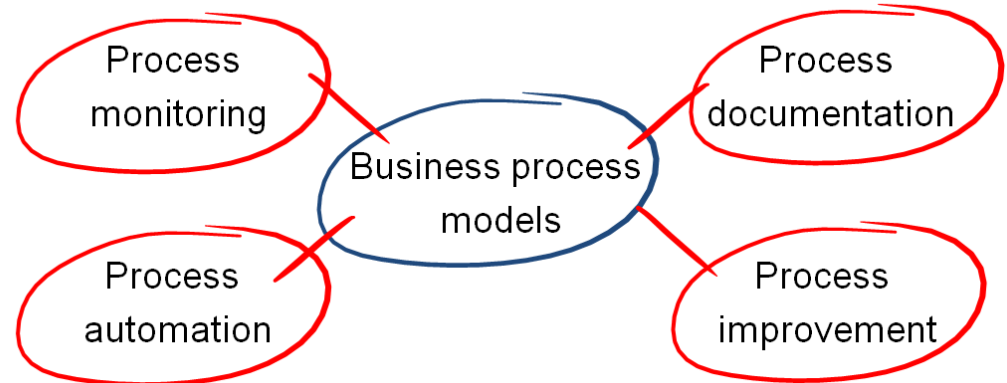
Simulating Business Processes

BPMN Meets DMN: Business Process and Decision Modeling:
An openHPI Course by Mathias Weske

Simulating Processes

• Motivation

- Study process executions without implementing the process for validation
- When we discuss or review process models, we „simulate“ them



Simulating Business Processes

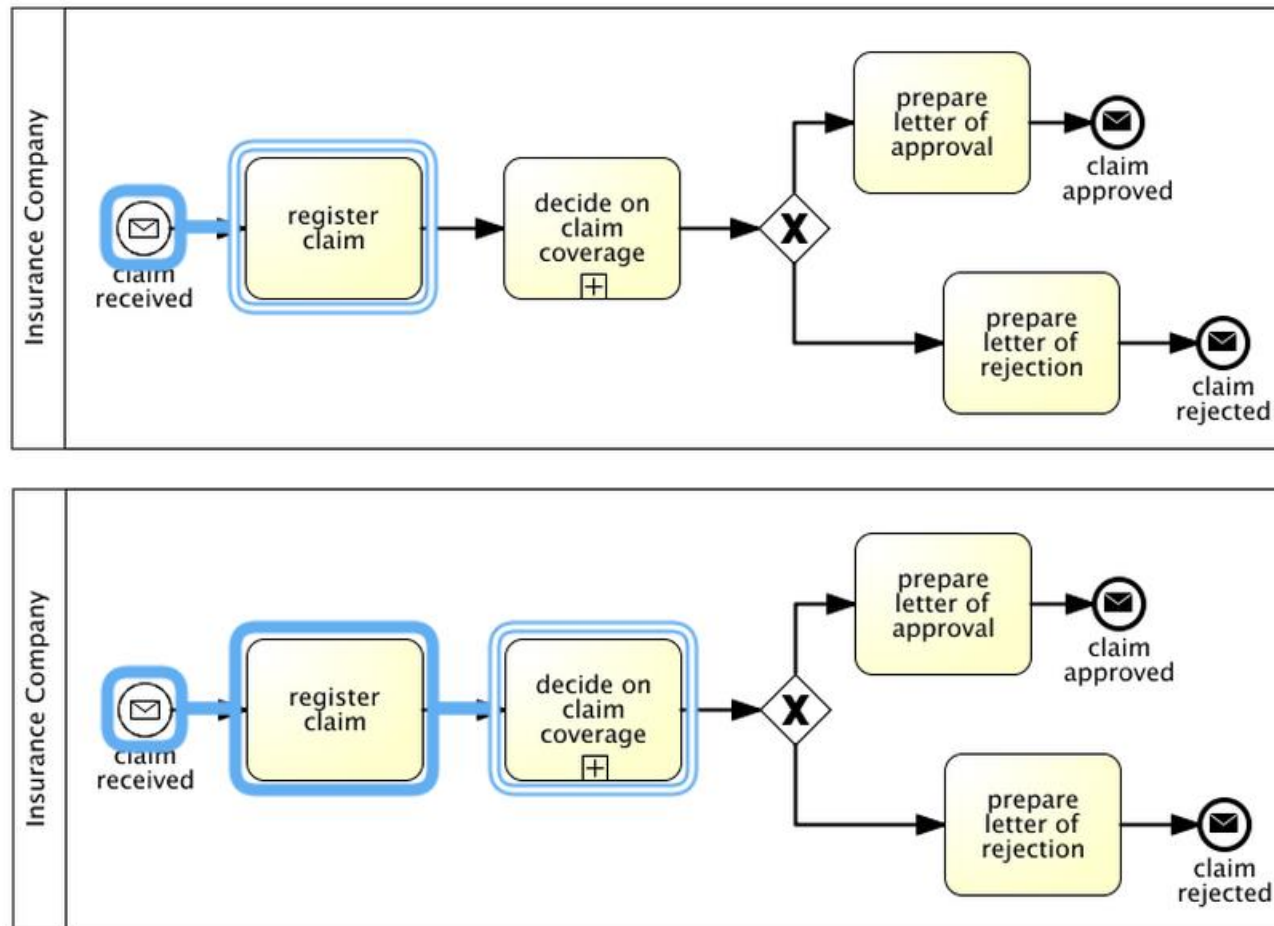
- Step-through simulation
 - Understand process behavior (qualitative simulation)
- Single-instance simulation
 - Associate additional properties to activities and resources and run a single instance to learn about its performance (e.g., cost)
 - Quantitative evaluation, one instance
- Multi-instances simulation
 - Run many process instances concurrently and get an idea about the runtime and cost of executing these process instances
 - Detailed representation of resources required
 - Not covered in this course

Step-through Simulation

- Usage
 - Used during process model design to achieve a common understanding of the behavior of a business process
 - Walk through a process model step by step
- Technically
 - Simulation is done by persons, using a simulation engine
 - Simulation engine calculates the dependencies of activities and triggers state transitions
- During step-through simulation, users trigger
 - state transitions of activity instances from ready to terminated, and
 - the selection of conditions

Step-through Simulation

- The user triggers a state transition from ready to terminated of the *register claim* activity instance



Step-through Simulation

- During simulation, we abstract from the actual running state
 - Which is appropriate, since we are interested in the logical ordering of activities, not in the activities themselves

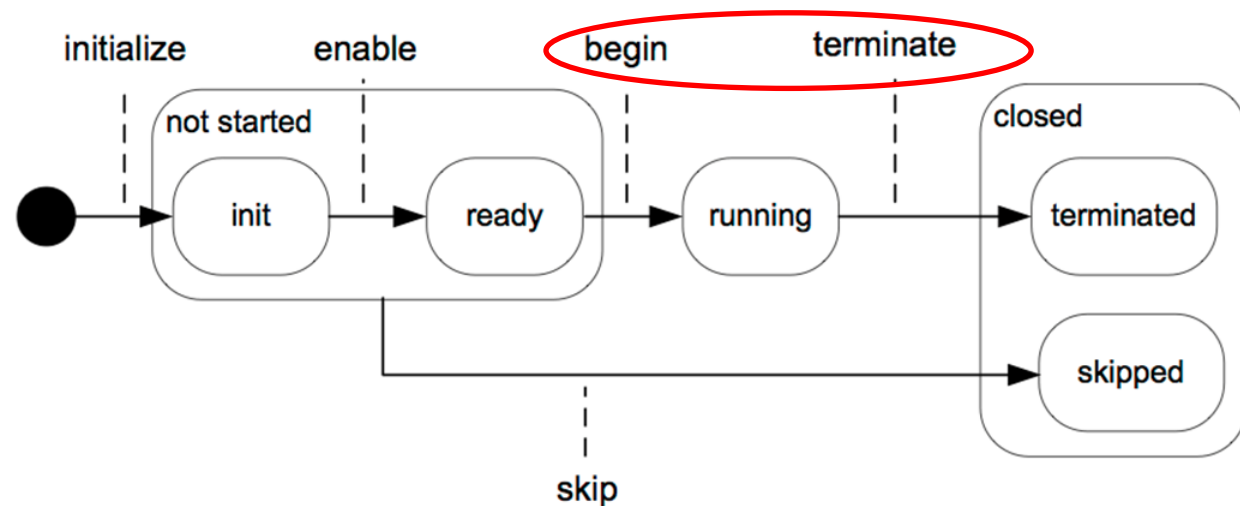
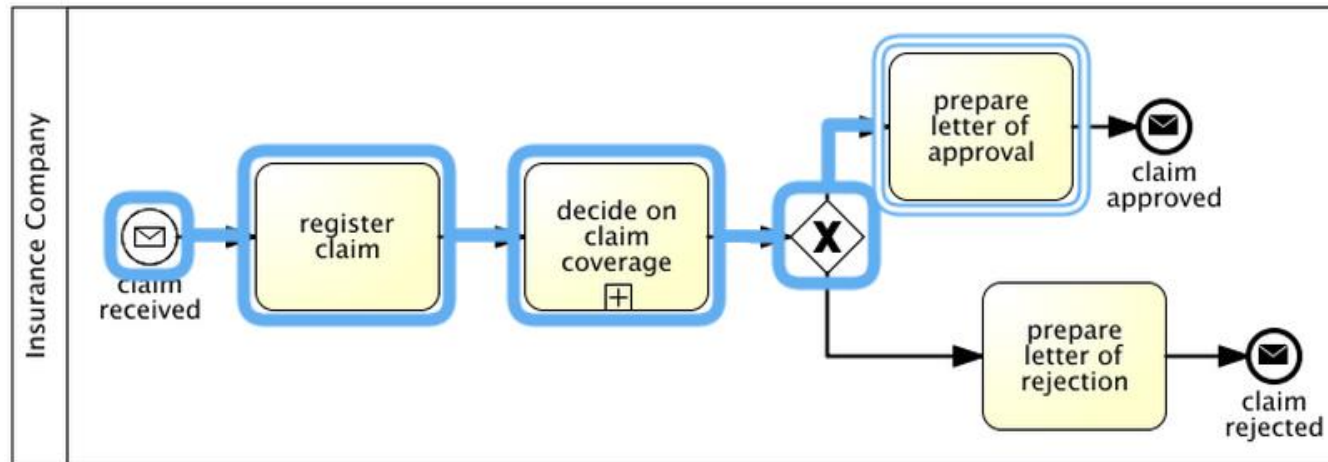
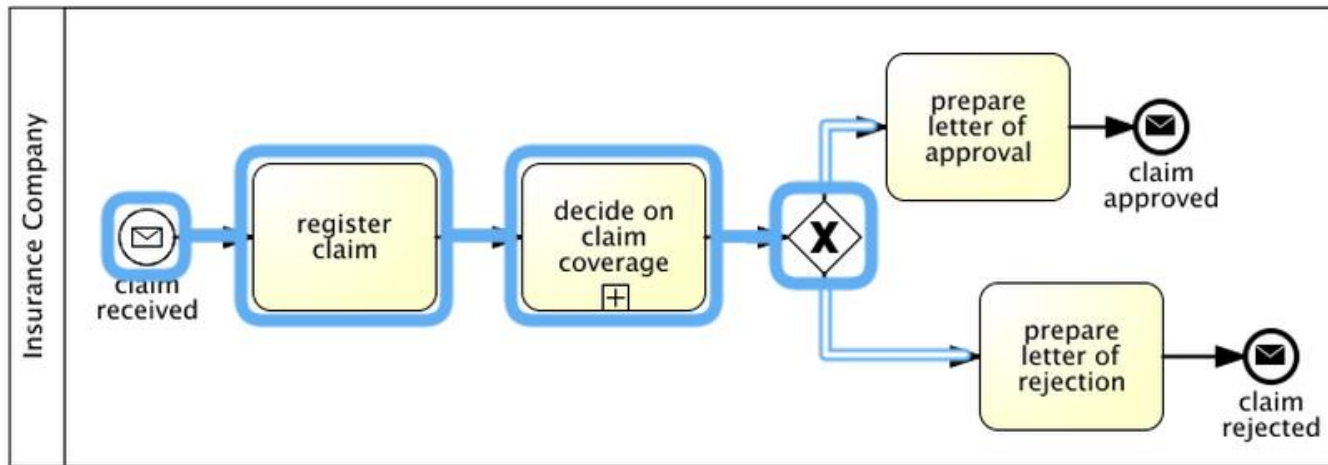


Fig. 3.9. State transition diagram for activity instances

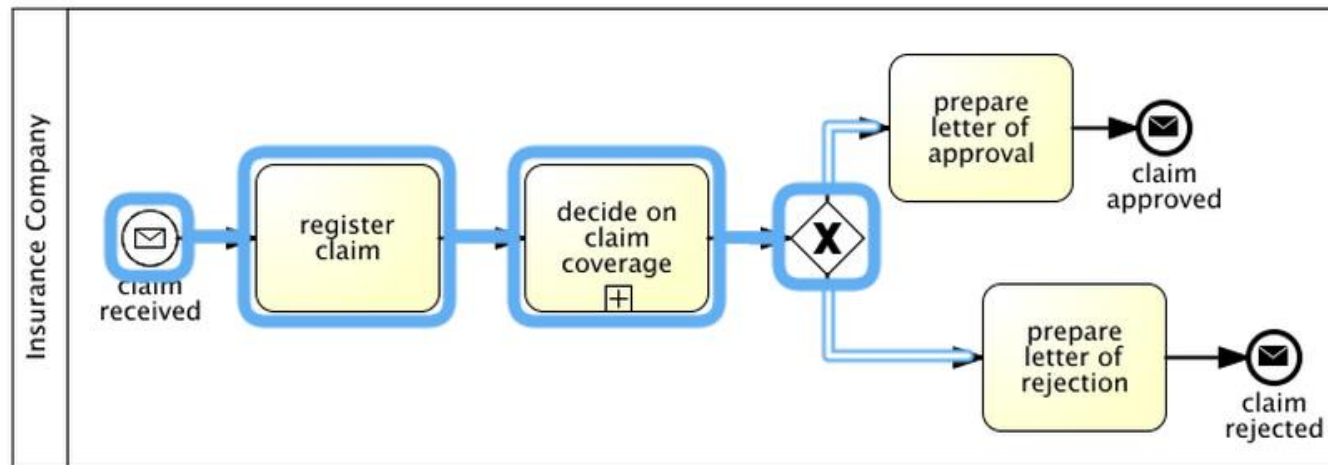
Step-through Simulation

- User selects a condition of an XOR split



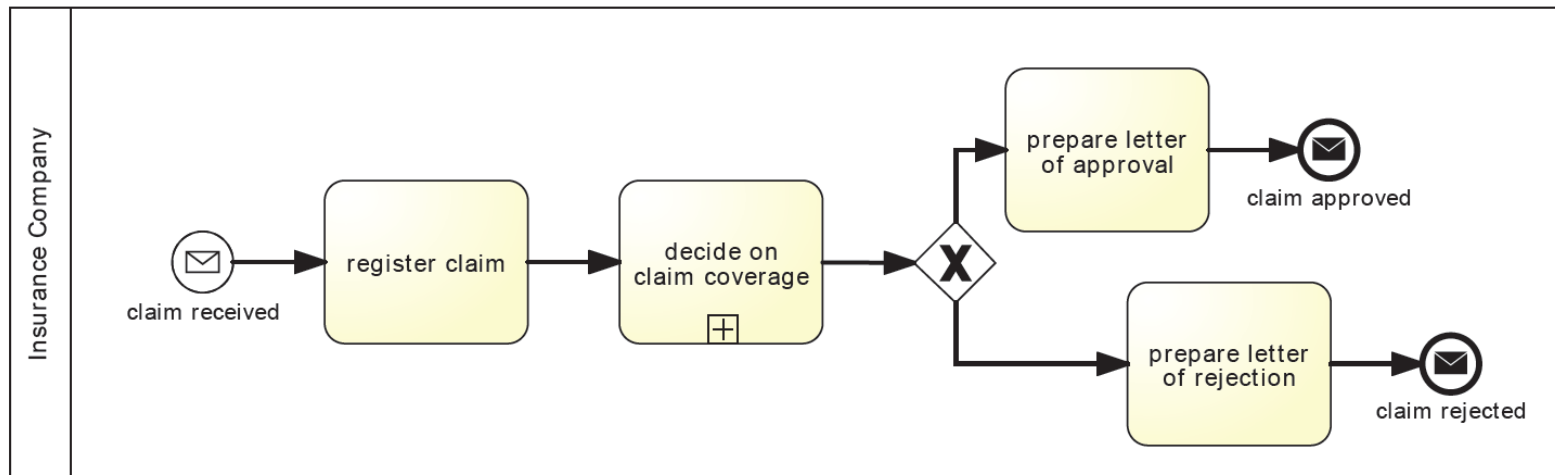
Quantitative Simulation

- Quantitative simulation allows estimating the performance of a process before its implementation
 - How long does a process instance take?
 - What is the cost of one process instance?
- Additional simulation information is required, for instance
 - Time and cost of conducting activities



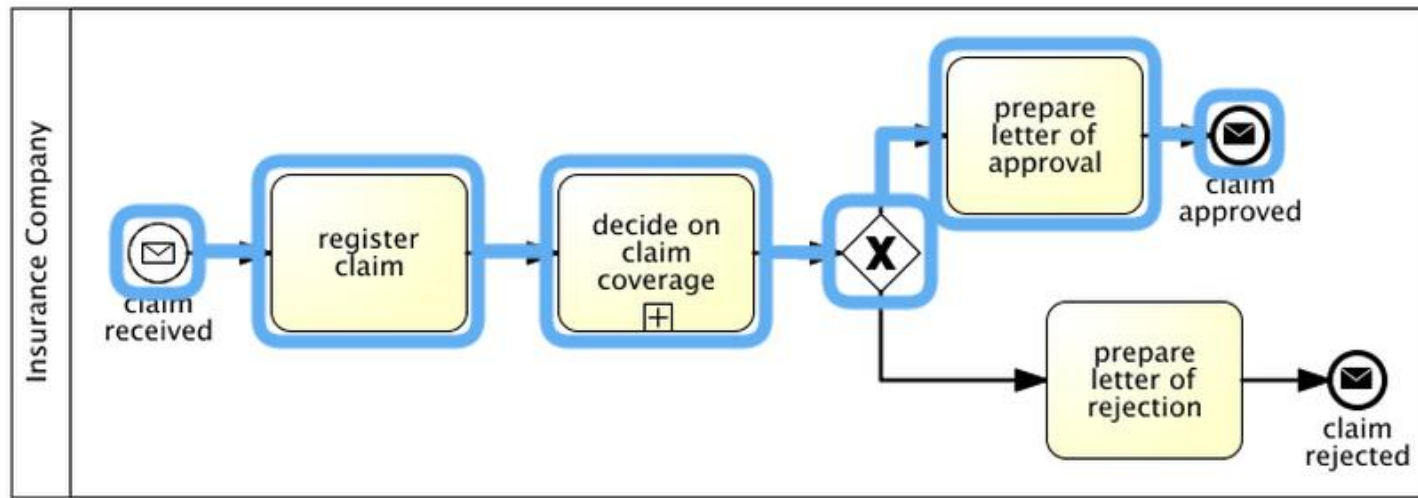
Simulating Insurance Claim Process

- Insurance claim scenario: compare costs to approve or reject a received claim
 - Activity processing times
 - register claim: 5 min
 - decide on claim coverage: 15 min
 - prepare approval letter: 8 min
 - prepare rejection letter: 15 min
 - Cost
 - Staff member: 40 €/hour, each letter sent: 1.5 €



Single Instance Simulation

- Cost and time for approving a claim



CURRENT RUN
(insurance claim)

Costs

€20.17

[more...](#)

Total cycle time

00:28h

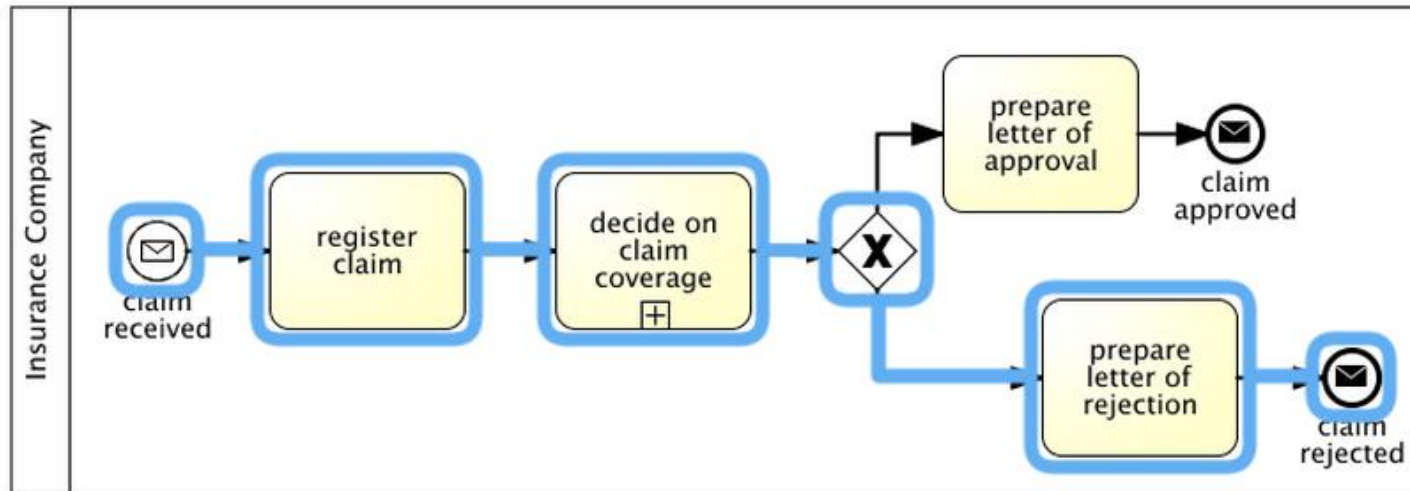
[more...](#)

- Cost calculation

- register claim: $5 \text{ min} \cdot 40 \text{ €/60 min} = 3.33 \text{ €}$
- decide on claim coverage: $15 \text{ min} \cdot 40 \text{ €/60 min} = 10 \text{ €}$
- prepare approval letter: $8 \text{ min} \cdot 40 \text{ €/60 min} = 5.33 \text{ €} + 1.5 \text{ €}$

Single Instance Simulation

- Cost and time for rejecting a claim



CURRENT RUN
(insurance claim)

Costs

€24.83

[more...](#)

Total cycle time

00:35h

[more...](#)

- Rejection is more costly, because writing of the letter takes longer
 - register claim: $5 \text{ min} \cdot 40 \text{ €/60 min} = 3.33\text{€}$
 - decide on claim coverage: $15 \text{ min} \cdot 40 \text{ €/60 min} = 10\text{€}$
 - prepare letter of rejection: $15 \text{ min} \cdot 40 \text{ €/60 min} = 10\text{€} + 1.5\text{€}$

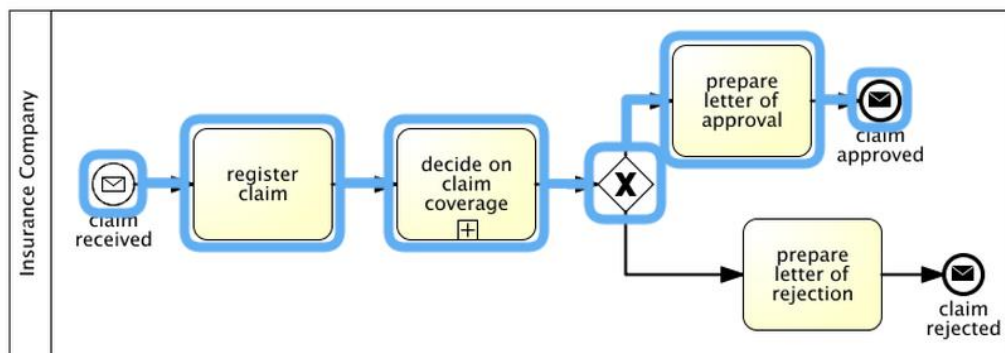
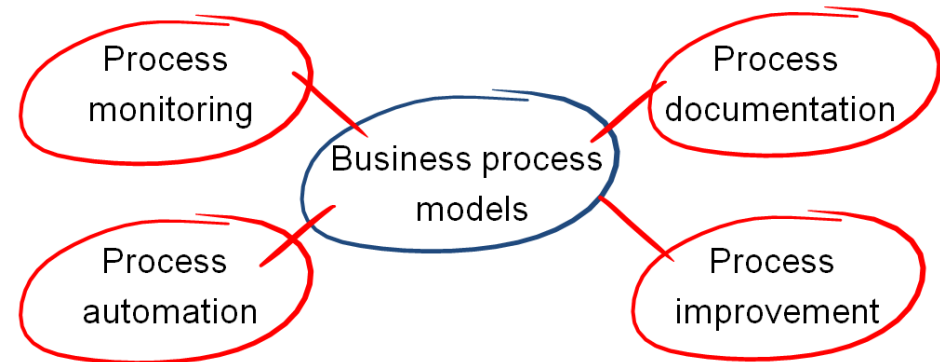
Multi-Instance Simulation

- Additional value comes with multi-instance simulation, which requires a more detailed simulation model
 - Case frequency (arrival times)
 - Available resources
 - Organizational structures, i.e., number of employees, equipment
 - Working times of employees
 - Start and end of simulation
 - Number of cases
 - Time interval
- Simulation runs a number of instances and aggregates cost and performance values
 - Shows bottlenecks, over- and underutilization of resources

Video Clip 3.4

Simulating Business Processes

- Evaluate process models without implementing them
- Step-through simulation for validation
- Quantitative analysis for performance evaluation



CURRENT RUN (insurance claim)	
Costs	
€20.17	more...
Total cycle time	
00:28h	more...

Video Clip 3.5

Informal Petri Net Primer

(Bonus Video)

BPMN Meets DMN: Business Process and Decision Modeling:
An openHPI Course by Mathias Weske

Petri Nets

- Petri nets are graphs that capture the behavior of systems
 - Formal representation of concurrent systems
- Background
 - Devised by Carl Adam Petri in 1962
 - Provide a theory for the communication of systems
 - Many extensions and variants have been developed since
 - Used in academic BPM to formally analyze process models



Petri Nets

- A Petri net is a directed graph comprised of places, transitions, and directed arcs between these
 - bipartite: arcs are only allowed between nodes of different types
- Terminology
 - p1 is input place of t1
 - p3 and p4 are output places of t2

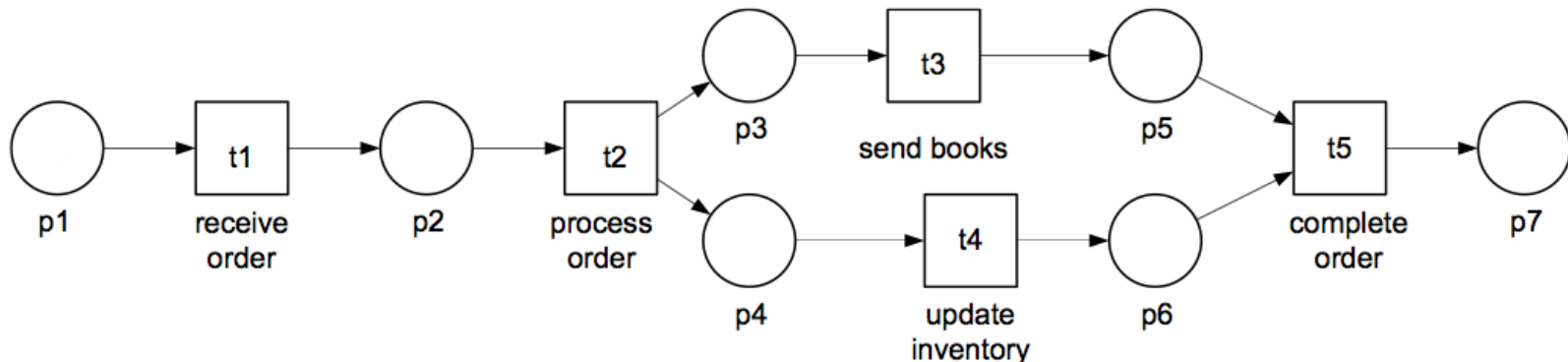


Fig. 4.26. Sample Petri net representing single process instance

Behavior of a Petri Net

- The behavior of a system is represented by token play in the Petri net
 - Each place may carry a number of tokens
 - The state of a Petri net is defined by the distribution of tokens over the places of the net
- State of the Petri net: [p1]

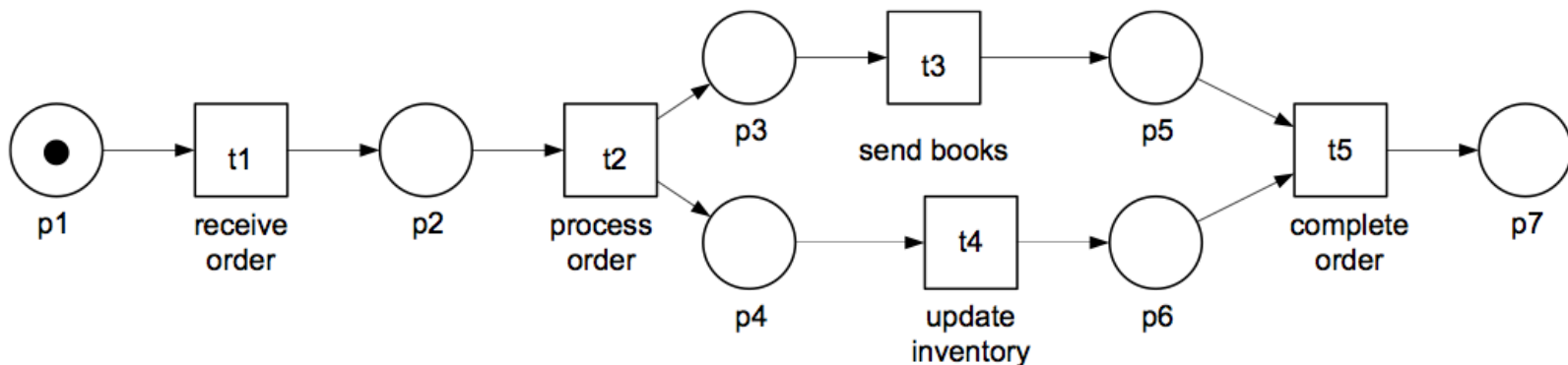


Fig. 4.26. Sample Petri net representing single process instance

Behavior by State Changes

- A Petri net changes its state by firing of a transition
 - Transition t can fire if it is enabled
 - t is enabled, if each input place carries at least one token
 - Firing of a t consumes one token from each input place of t and produces one token on each output place of t

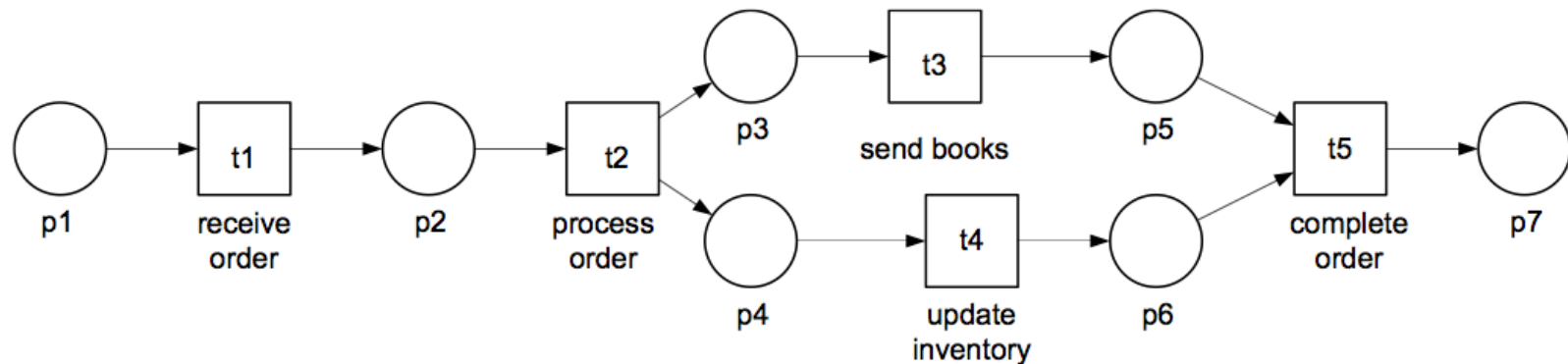


Fig. 4.26. Sample Petri net representing single process instance

Reachability

- Applying all possible state changes to a Petri net results in its reachability graph

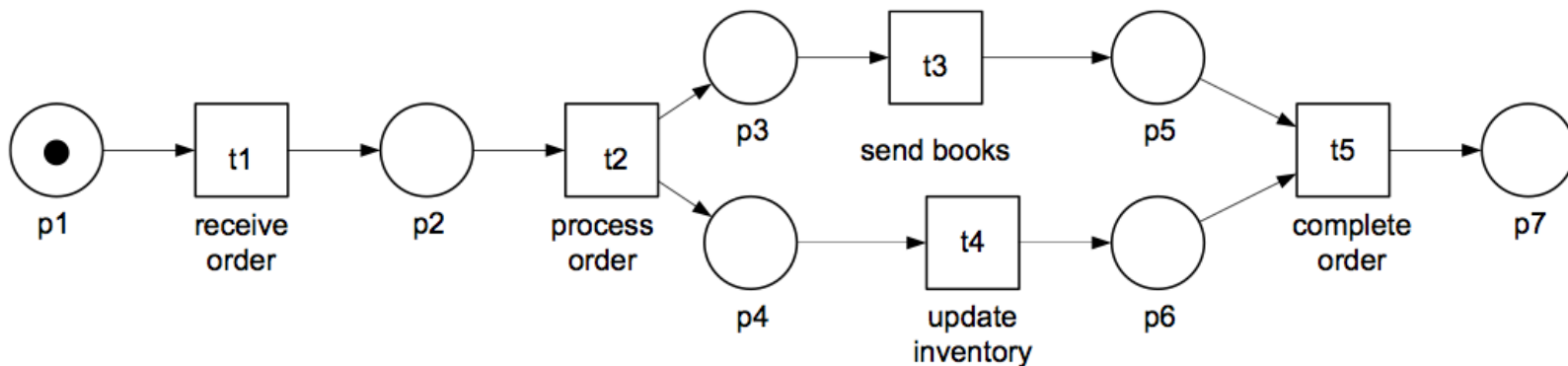
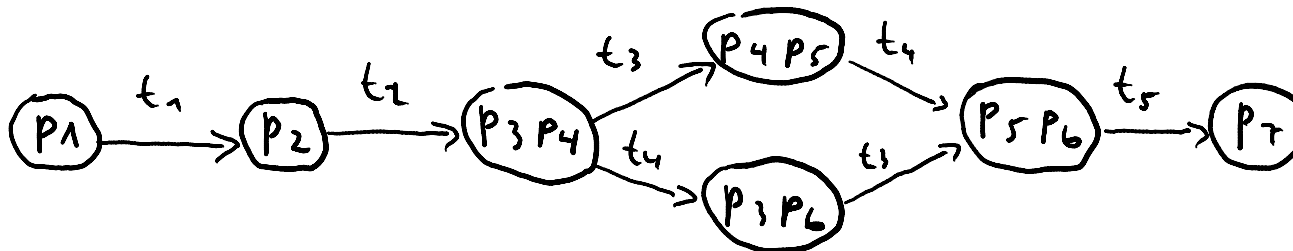


Fig. 4.26. Sample Petri net representing single process instance

Video Clip 3.5

Petri Net Primer

- Petri net introduction
- Structure of a Petri net
- Behavior of a Petri net
- Reachability graph

Video Clip 3.6

Mapping BPMN to Petri Nets

(Bonus Video)

BPMN Meets DMN: Business Process and Decision Modeling:
An openHPI Course by Mathias Weske

Petri nets and Soundness

- Idea
 - Use a reachability graph to determine soundness of a Petri net
 - But we have modeled processes in BPMN
- Next steps
 - Provide a mapping from (a subset of) BPMN to Petri nets
 - Check soundness properties (1), (2) and (3), using the reachability graph of the corresponding Petri net
- A Petri is structurally sound, if
 - one dedicated initial place i that has no incoming arcs,
 - one dedicated final place o that has no outgoing arcs,
 - each node of the net lies on a directed path from i to o .
 - In this case, it is called a workflow net

Translating Business Processes to Petri Nets

- Idea
 - BPMN activities, events, and gateways are mapped to Petri net subnets
 - These subnets are connected such that they represent the behavior of the business process model
- Please notice
 - Intermediate events and tasks are mapped to transitions
 - Bipartite structure of Petri nets requires additional transitions
 - Transitions that do not have an activity or intermediate event associated in the BPMN process model
 - These transitions are called silent transitions

Translating Process Activities

- Execution of an activity is represented by the firing of a transition



- We are only interested in the termination of activities

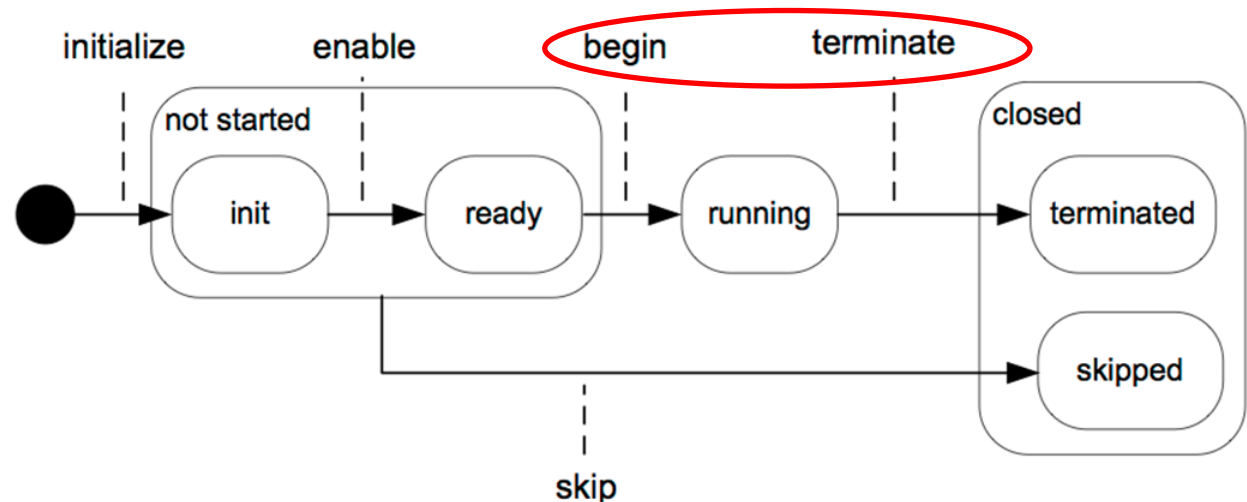
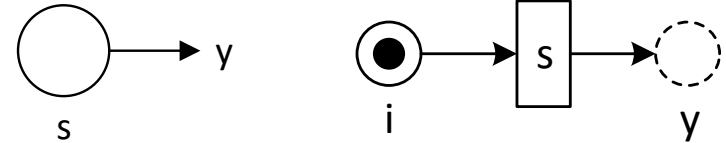


Fig. 3.9. State transition diagram for activity instances

Events

- Start event

- Is mapped to a place i with a token and a transition that represents the occurrence of the start event



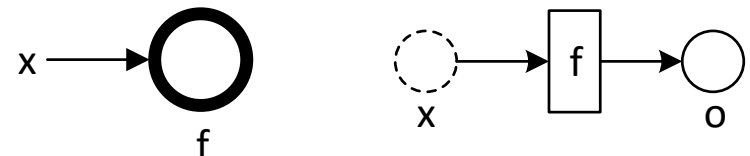
- Intermediate event

- We abstract from the kind of intermediate event and represent it by a transition



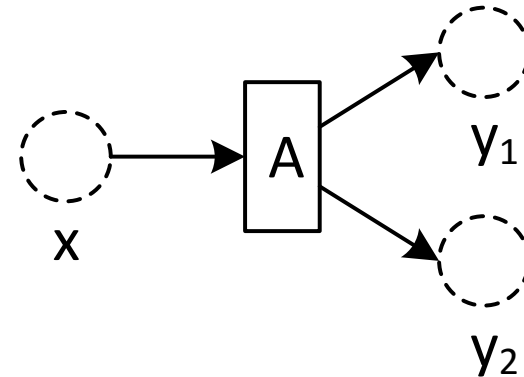
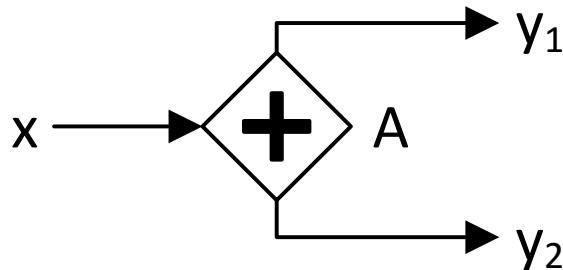
- End event

- Is mapped to a transition and a place o

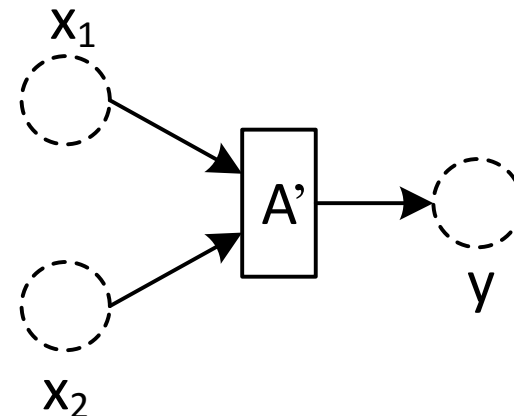
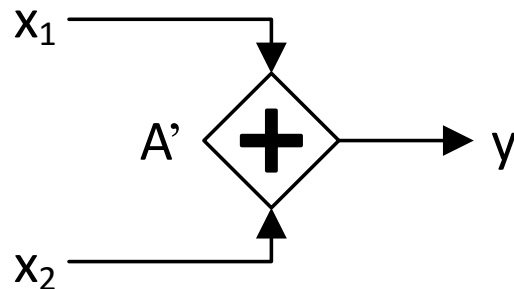


Parallel Gateway

- AND split
 - Uses the split behavior of transitions

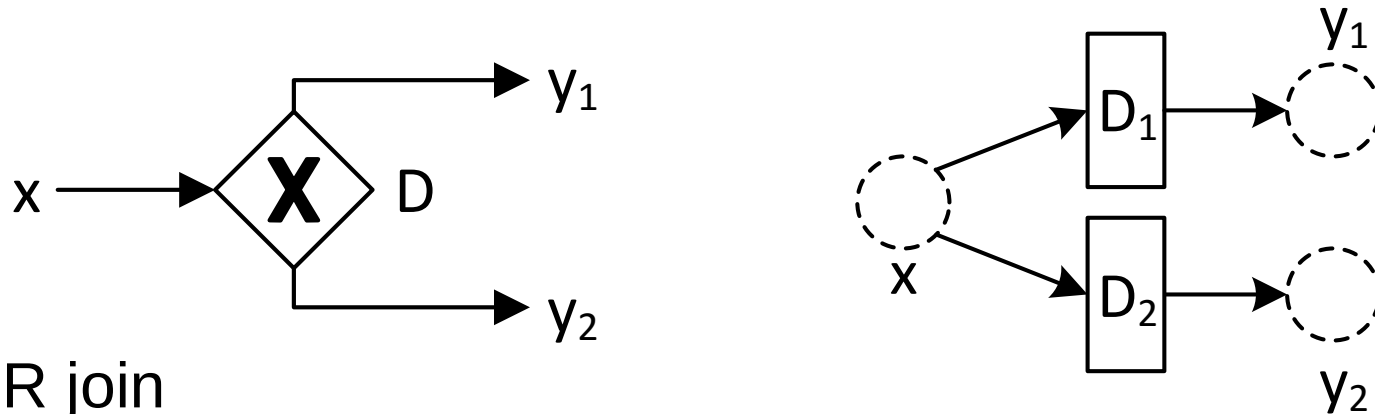


- AND join
 - Uses the join behavior of transitions

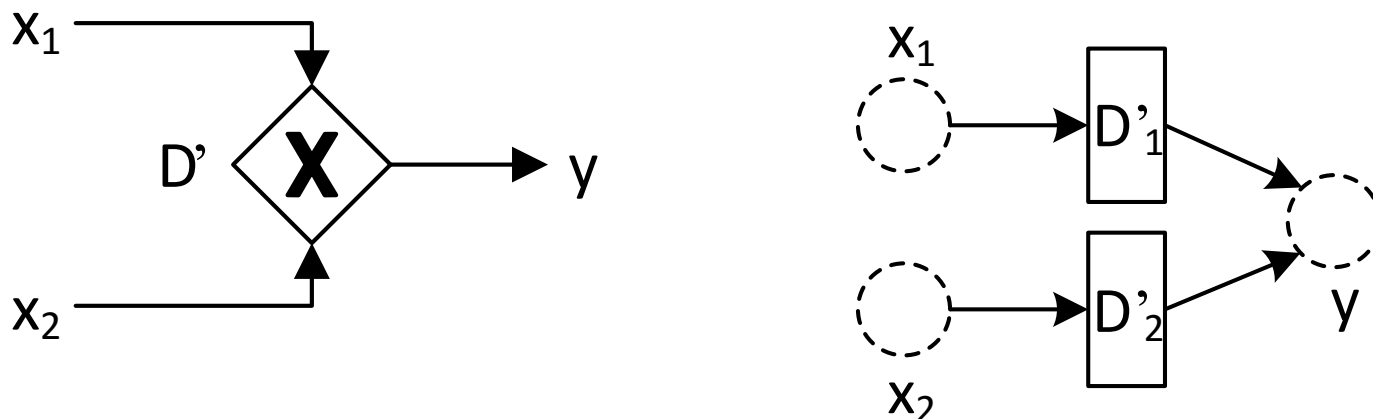


Exclusive Gateway

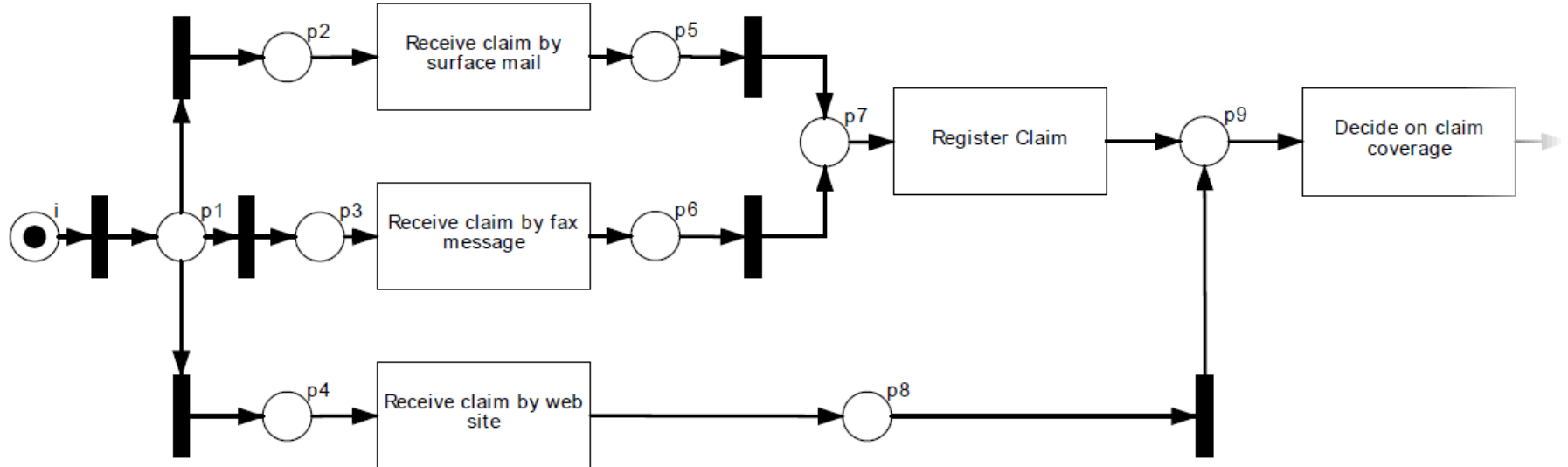
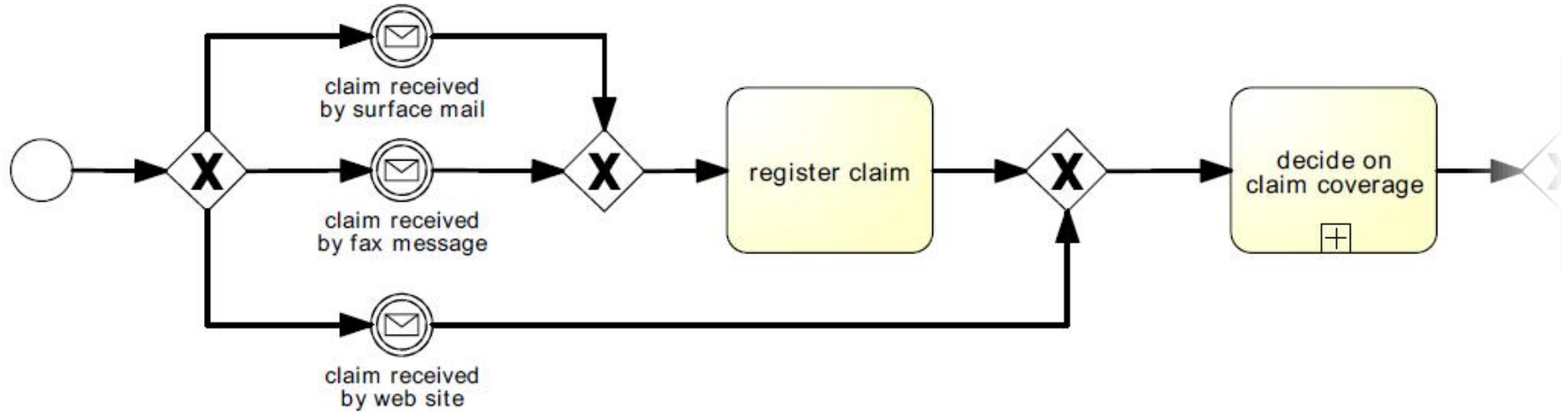
- XOR split
 - Introduces silent transitions to represent decisions



- XOR join
 - Uses silent transitions to represent the merge



Transform Process Model to Petri Net



Video Clip 3.6

BPMN to Petri Nets

- Translating
 - Activities,
 - Events, and
 - Gateways
- Example

Video Clip 3.7

Checking Soundness

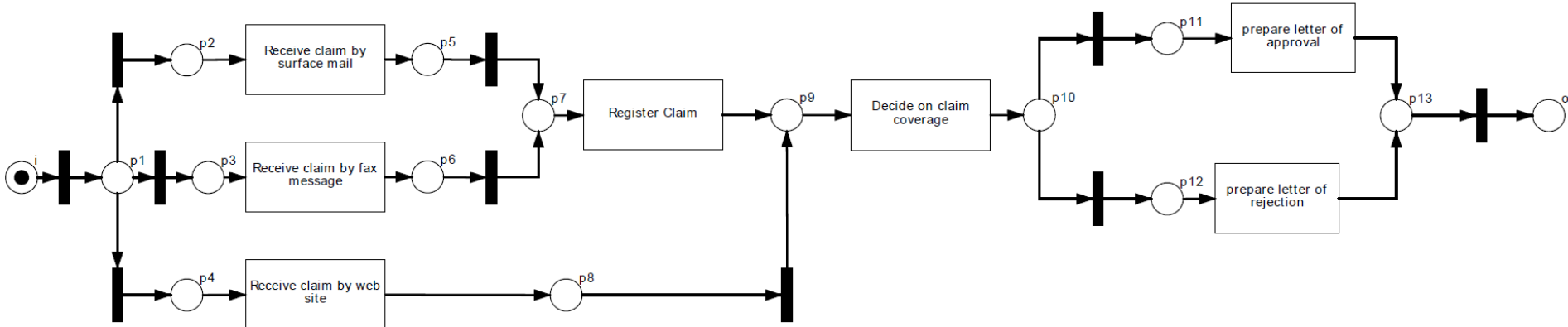
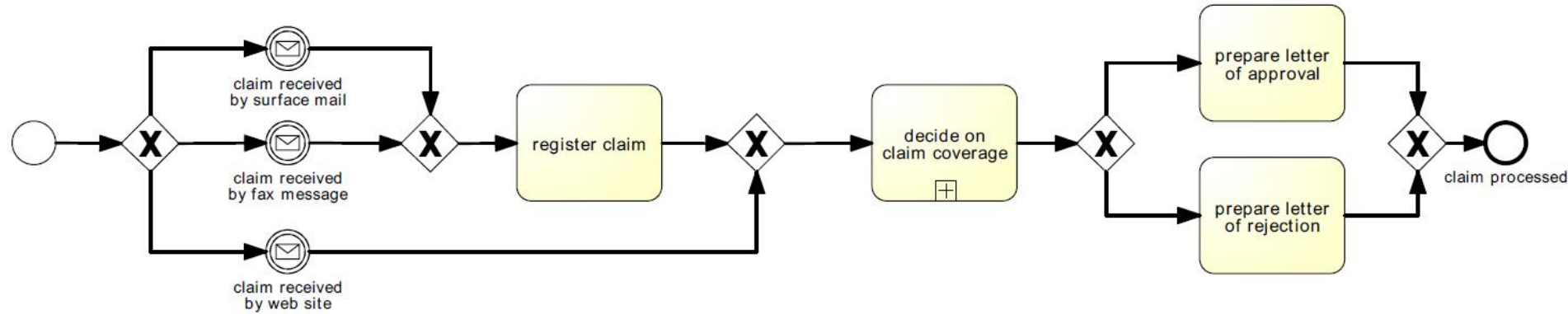
(Bonus Video)

BPMN Meets DMN: Business Process and Decision Modeling:
An openHPI Course by Mathias Weske

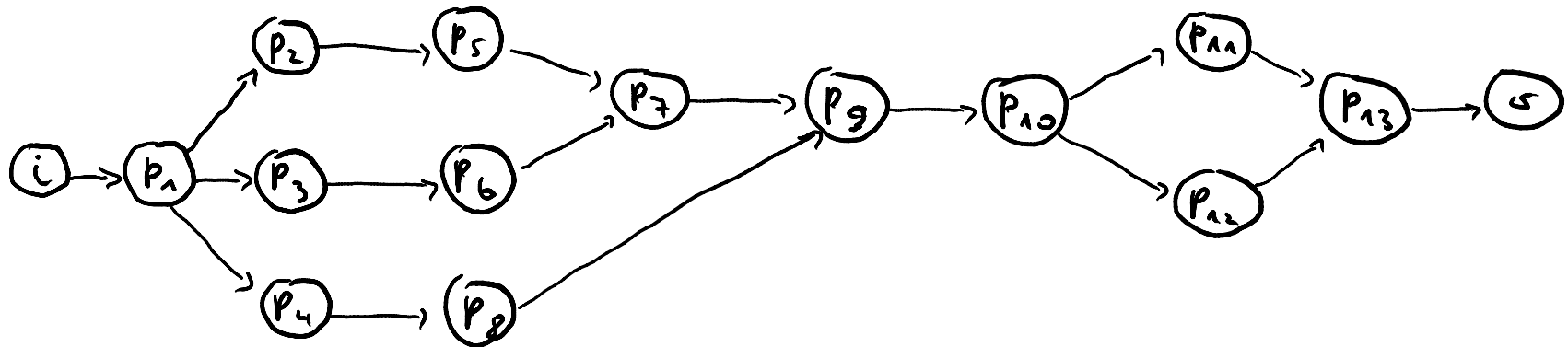
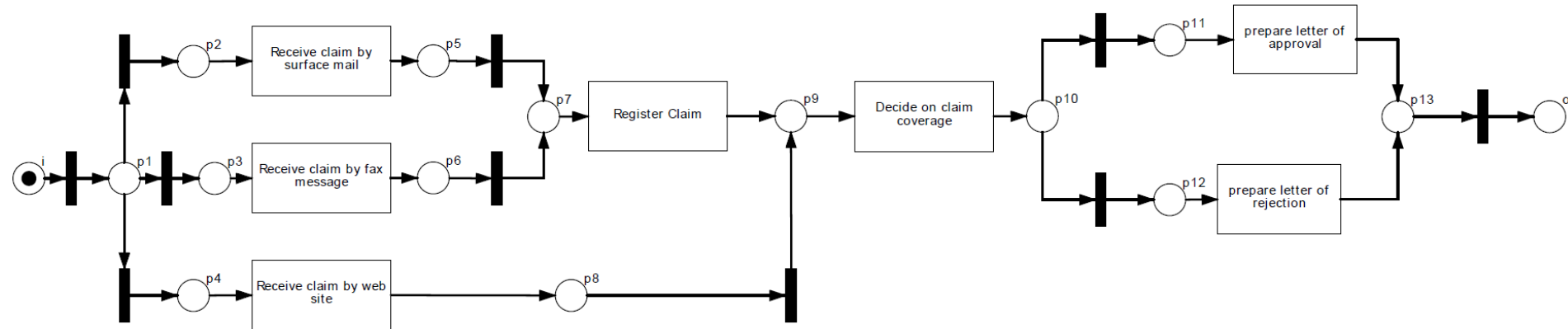
Checking for Soundness

- Transform process model to Petri net
- Create reachability graph of Petri net
- Use reachability graph to check soundness
 - (1) Starting in state i (one token in the net, on place i), all paths will reach a state in which there is one token on place o
 - (2) If a state is reached with a token on place o , all other places are empty
 - (3) For each transition t in the Petri net there is an arc in the reachability graph that is marked t
- Fairness Property
 - All decisions that a process can take will be taken eventually
 - Transitions do not starve

Translate Process Model to Petri Net

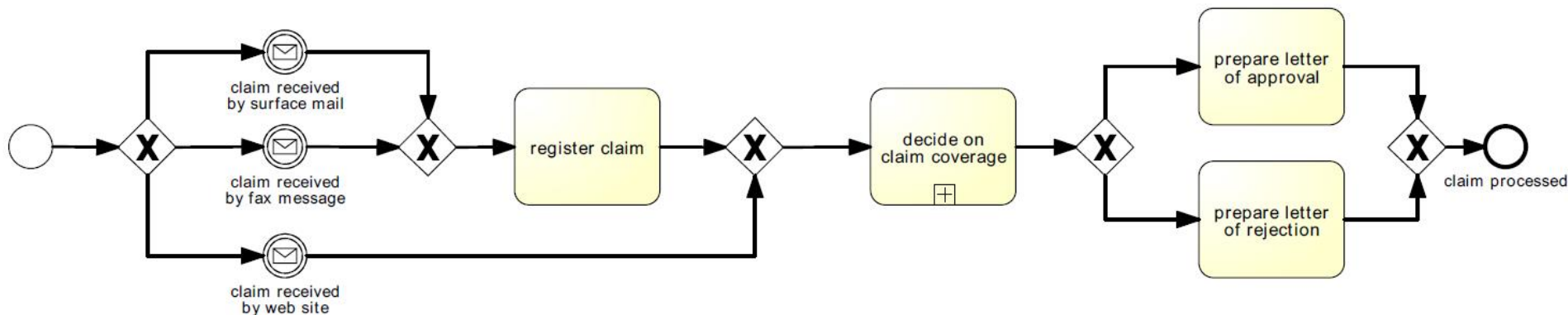


Building the Reachability Graph

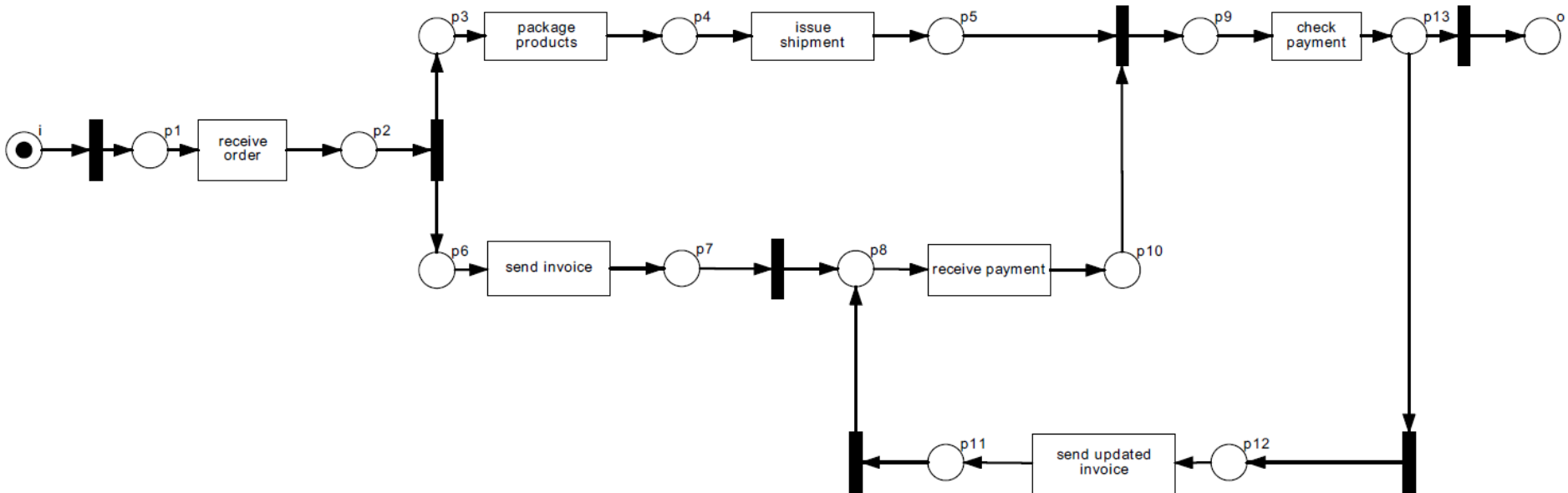
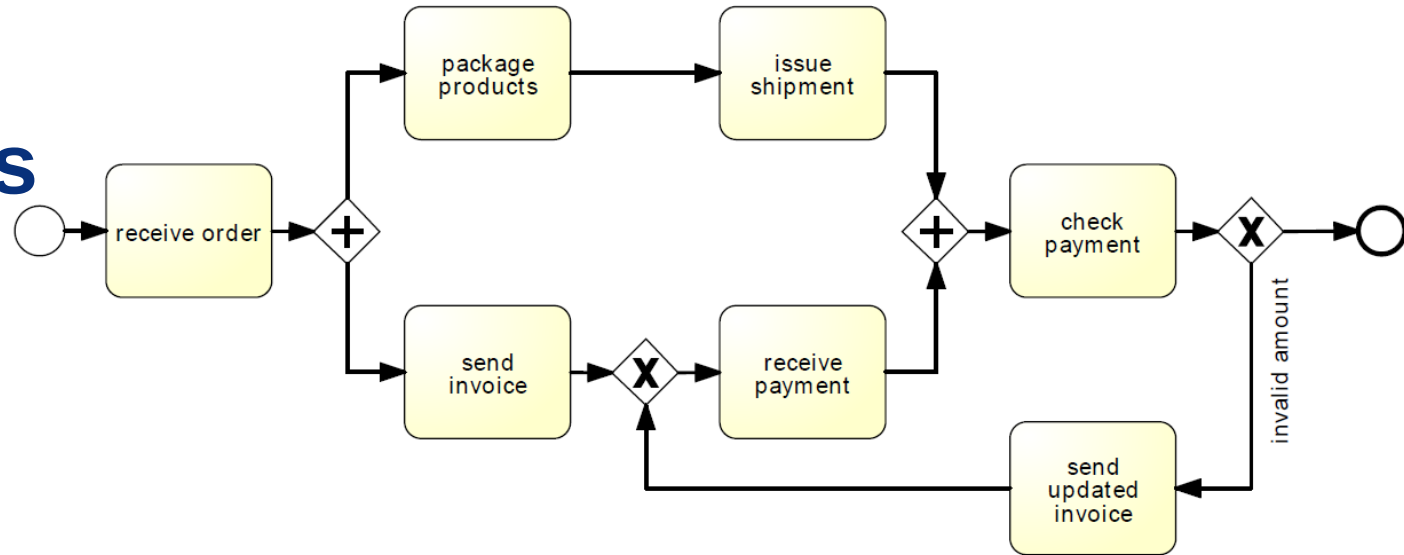


Checking for Soundness

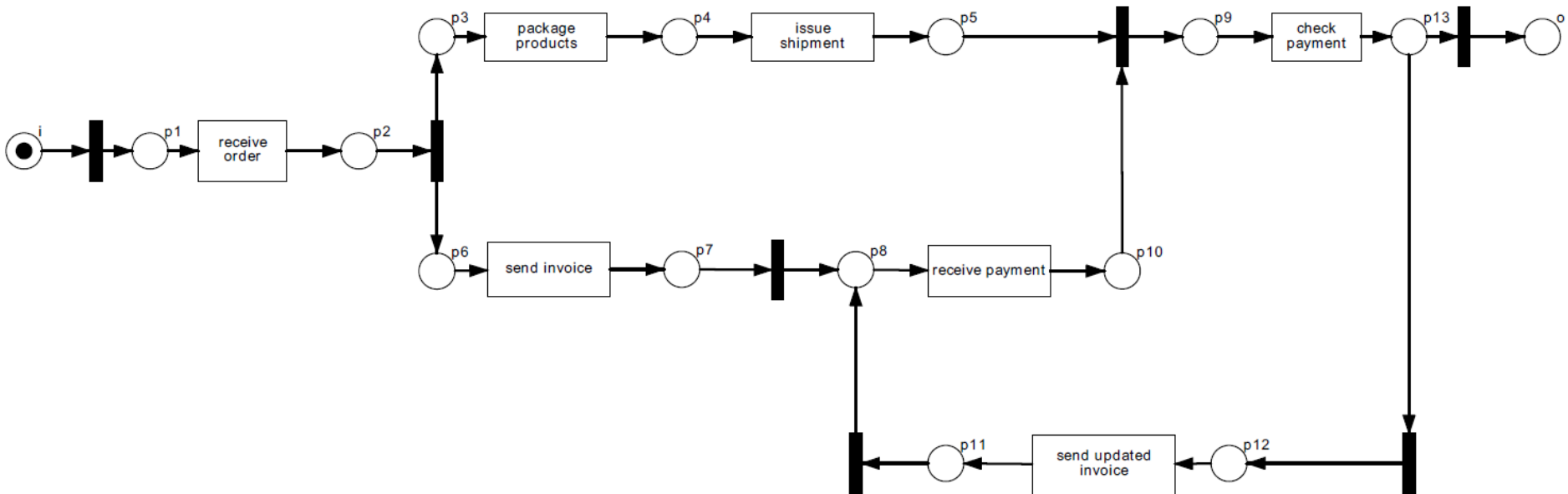
- Since properties (1), (2), and (3) are met, the Petri net is sound and so is the process model



Sample Soundness Analysis

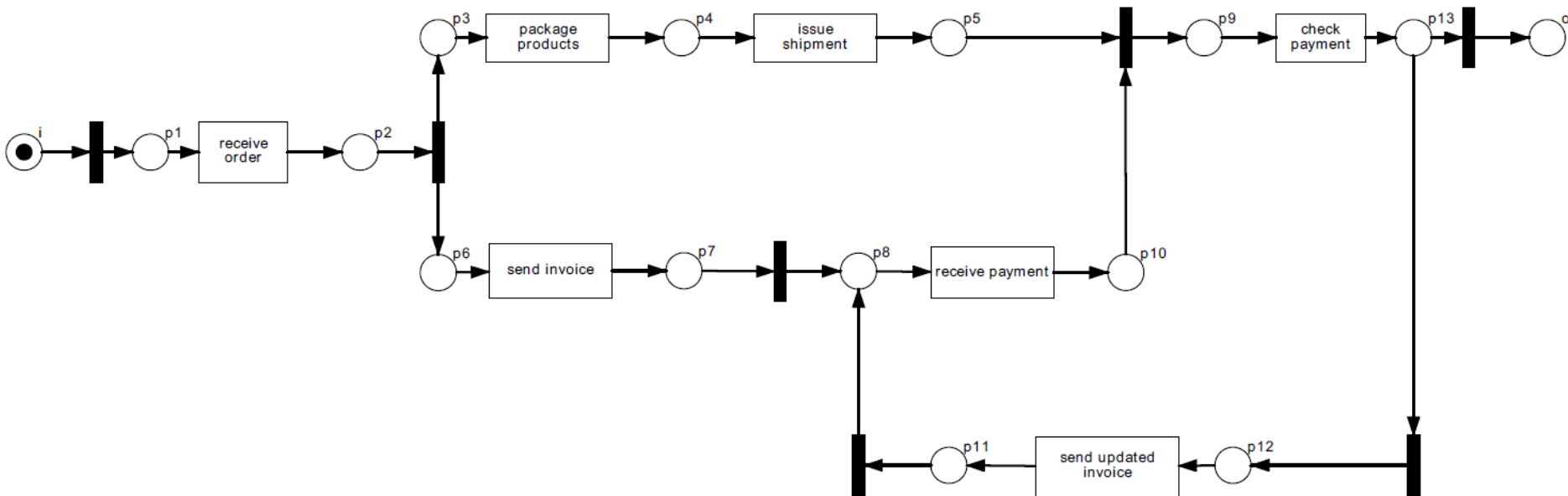


Building the Reachability Graph



Checking the Reachability Graph

- Since properties (1), (2), and (3) are violated, the Petri net is not sound
- The BPMN process model is also not sound



Service-technology.org

- Soundness checking can be done by software tools
- Service-technology.org provides powerful tools to analyze formal properties of services and business processes
 - Developed by academic partners from U Rostock, HU Berlin and TU Eindhoven
 - Available for download

SERVICE-TECHNOLOGY.ORG
SOLUTIONS THAT MAKE SERVICES BEHAVE WELL

Video Clip 3.7

Checking Soundness

- Translating a process model to a Petri net
- Building its reachability graph
- Checking it for soundness
- Software tools available

Summary of Week 3

- Process behavior
 - Execution semantics
 - Behavioral anomalies
 - Structural soundness and how to achieve it
 - Soundness as correctness criterion
- Simulation
 - Validate business processes using step-through
 - Analyze performance by quantitative simulation
- Soundness, more formally (Bonus videos)
 - Mapping of BPMN models to Petri nets
 - Reachability graph of Petri net
 - Use reachability graph to check soundness