Flexible Process Notations for Cross-organizational Business Process Management

Tijs Slaats

07-12-2015

Based on joint work with Thomas Hildebrandt, Raghava Rao Mukkamala and Soren Debois

eXformatics

IT University of Copenhagen

Overview

- Background
  - Business Process Management
  - Flexible Notations for Knowledge Work
- Dynamic Condition Response (DCR) Graphs
- DCR Graphs for Cross-organizational Processes
- Conclusion
- Demo

Business Process Management

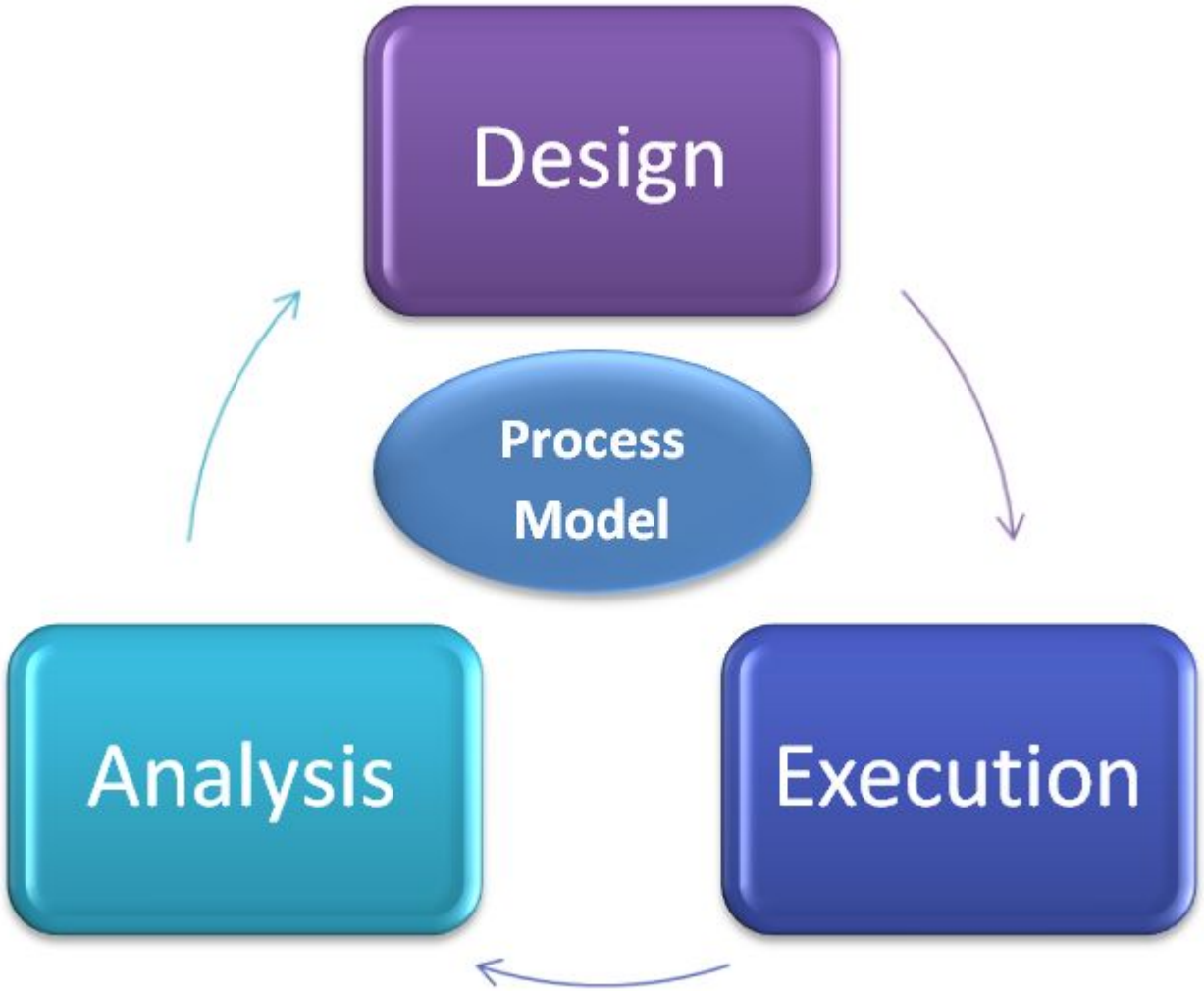
Business Process: "A structured, measured set of activities designed to produce a specific output for a particular customer or market"[1]

Examples:

- Production of car
- Handling of an insurance claim
- Treatment for lung cancer

[1] Thomas Davenport -Process Innovation: Reengineering work through information technology. (1993)

Business Process Management



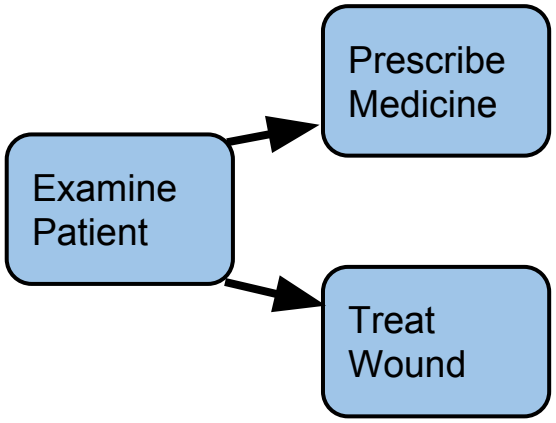
Business Process Modelling

How do we model a process?

- Plain text:

“Attach wheels and engine to frame.”

- Drawings:



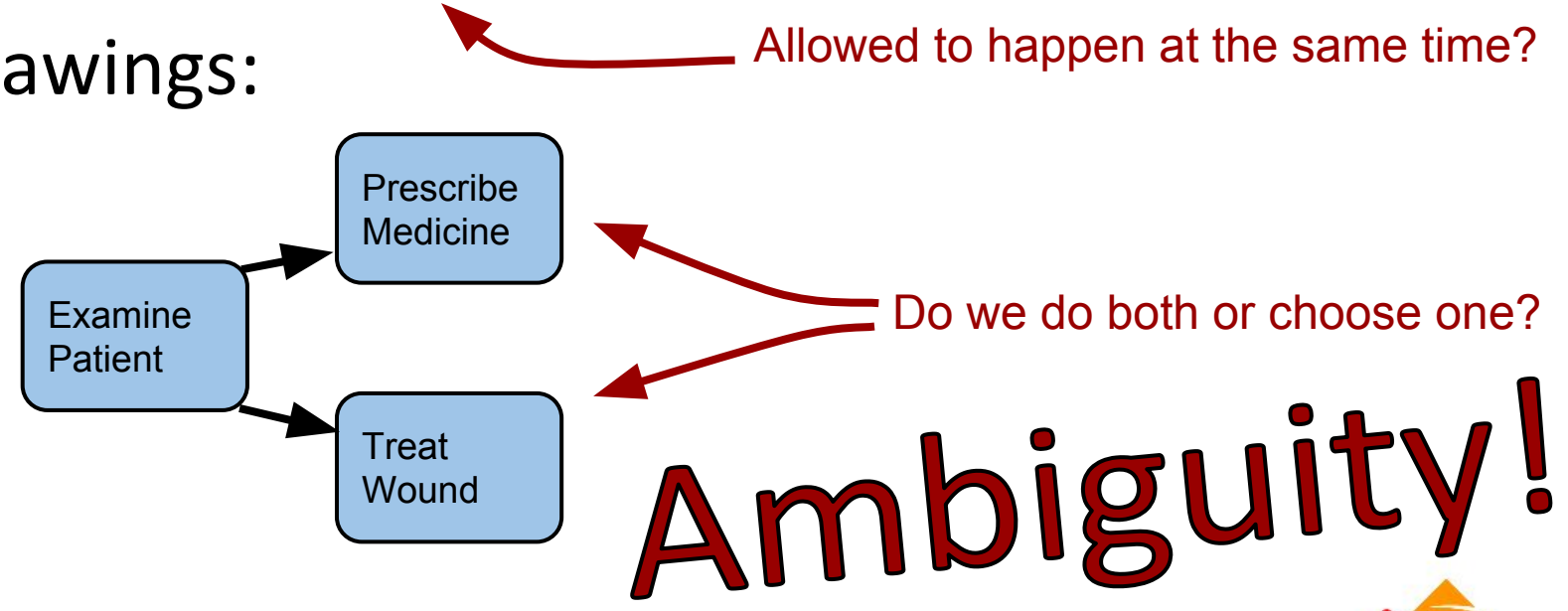
Business Process Modelling

How do we model a process?

- Plain text:

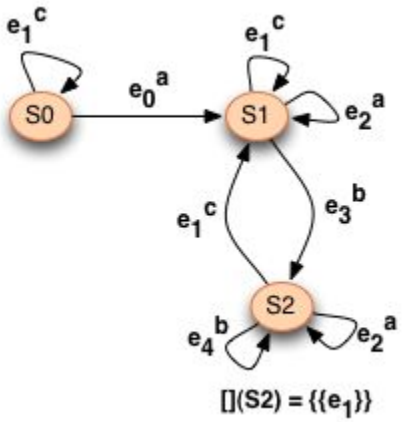
"Attach wheels and engine to frame."

- Drawings:

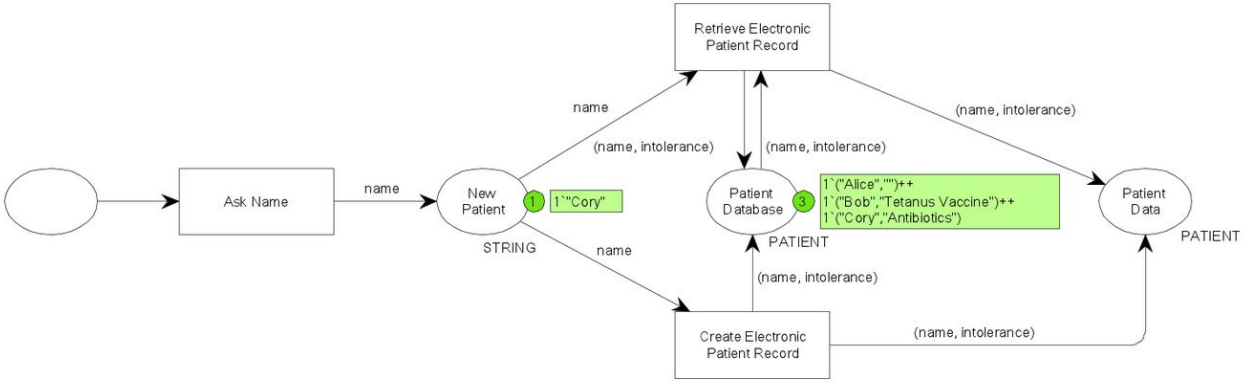


BPM and Computer Science

We need a well-defined language for our models: **Formal Methods**



$e_0 | e_1$, $e_1 | e_2$, $e_2 | e_3$, $e_2 | e_4$



□(Recieve Claim ⇒ ◇Evaluate Claim)
□(Approve Claim ⇒ ◇Payout Claim)
□(¬Payout Claim W Approve Claim)

eXformatics

IT University of Copenhagen

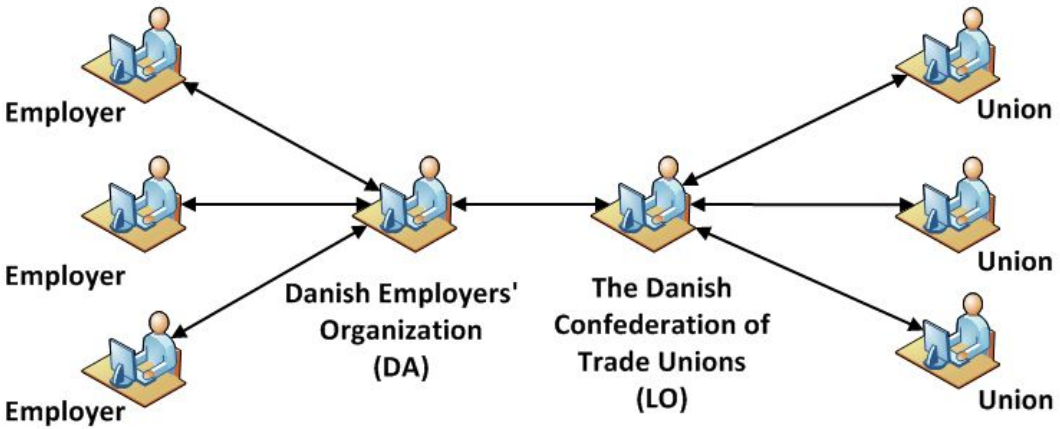
BPM and Computer Science

Formal models offer:

- Unambiguous semantics
- Verification
- Model checking
- Simulation
- Execution
  - Automated (fx assembly lines)
  - User guidance (fx call centers)

BPM and Distributed Systems

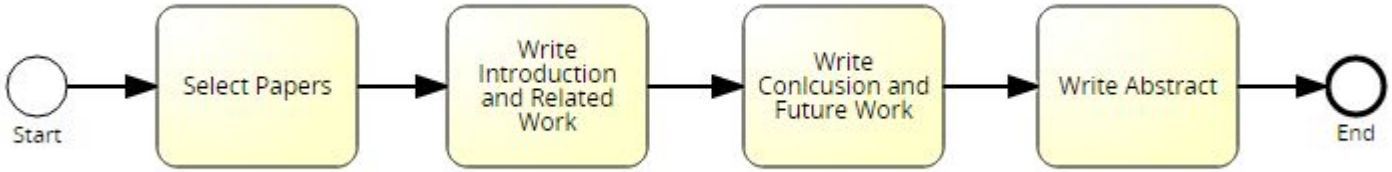
Business processes are often cross-organizational.



Business Process Modelling

Business Process Model Notation (BPMN):

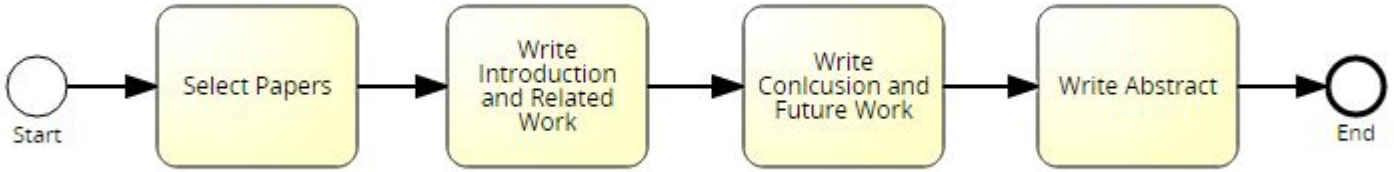
- Standard notation used by industry
- Has (to a large degree) been formalized
- Is essentially flow-based:



Business Process Modelling

Business Process Model Notation (BPMN):

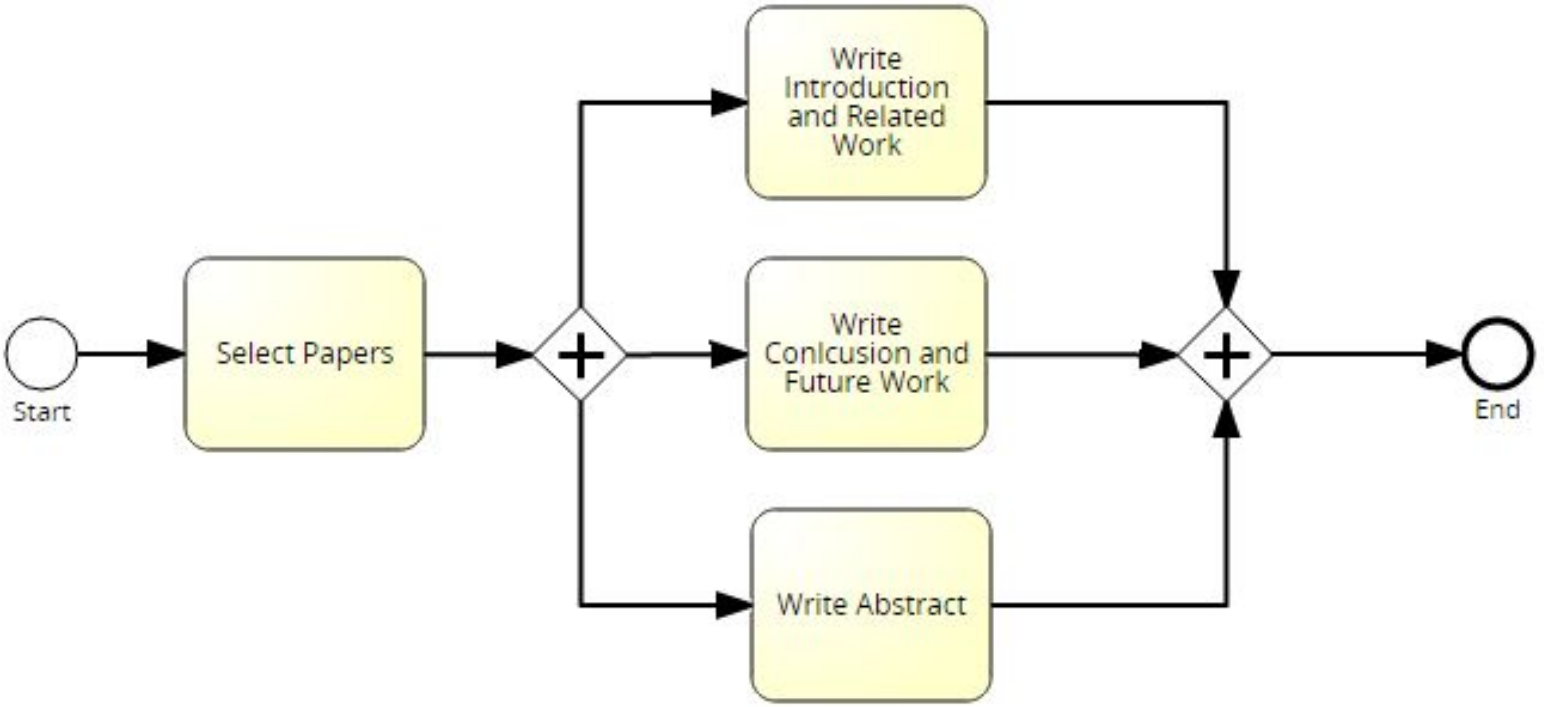
- Standard notation used by industry
- Has (to a large degree) been formalized
- Is essentially flow-based:



Nice for straightforward, strict processes, but what if we want something more flexible?

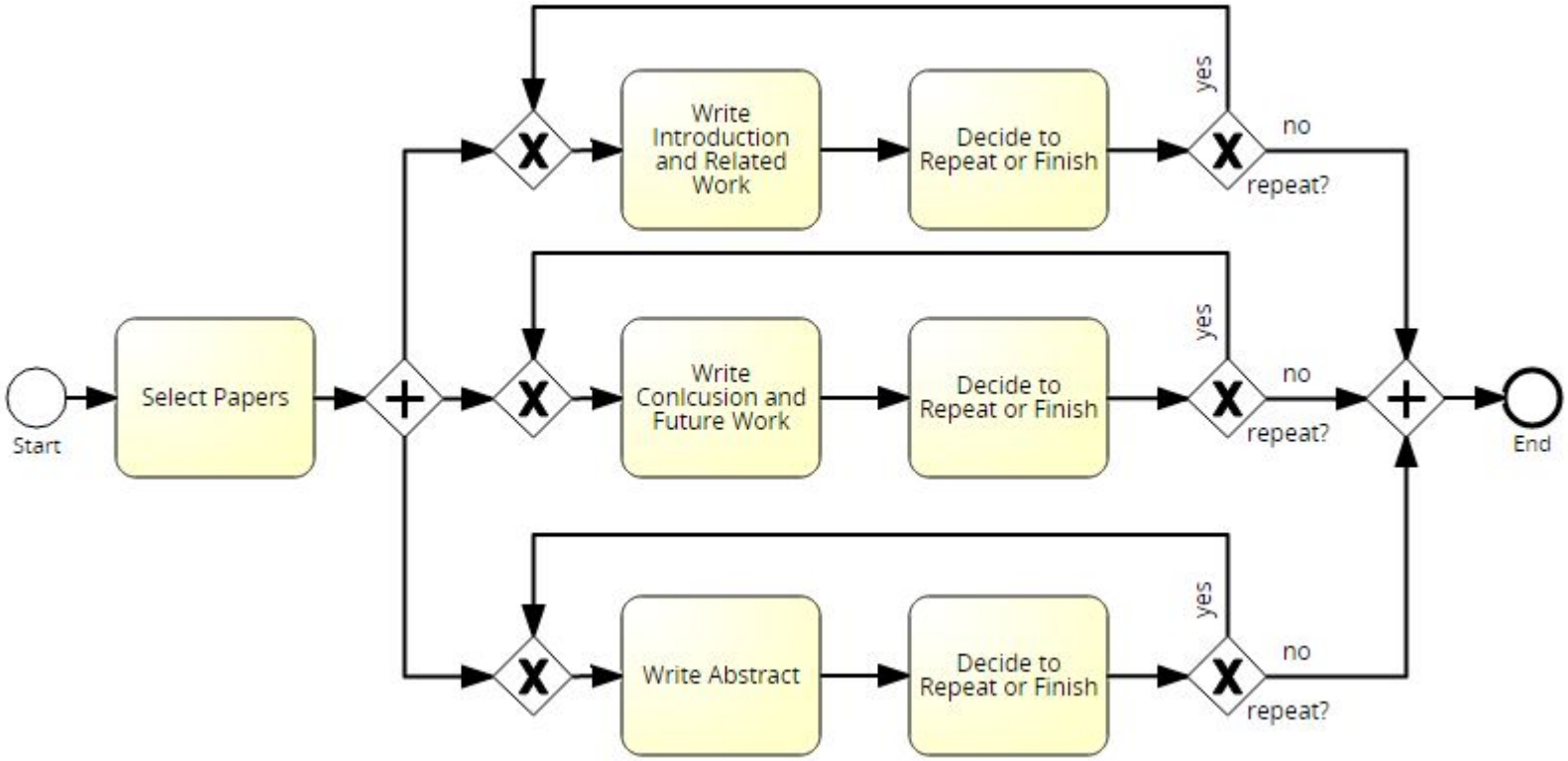
Business Process Modelling

What if we want to be able of choosing the order of activities?



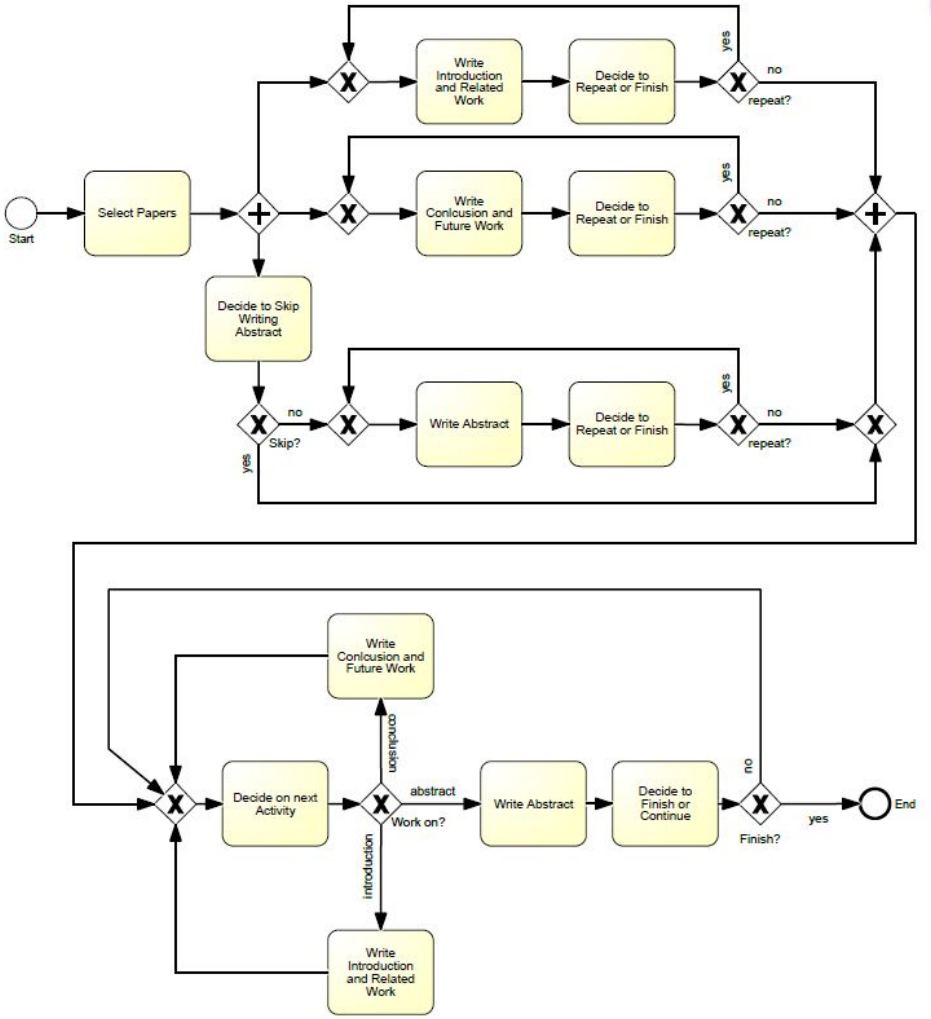
Business Process Modelling

What if we want to be able of repeating activities?



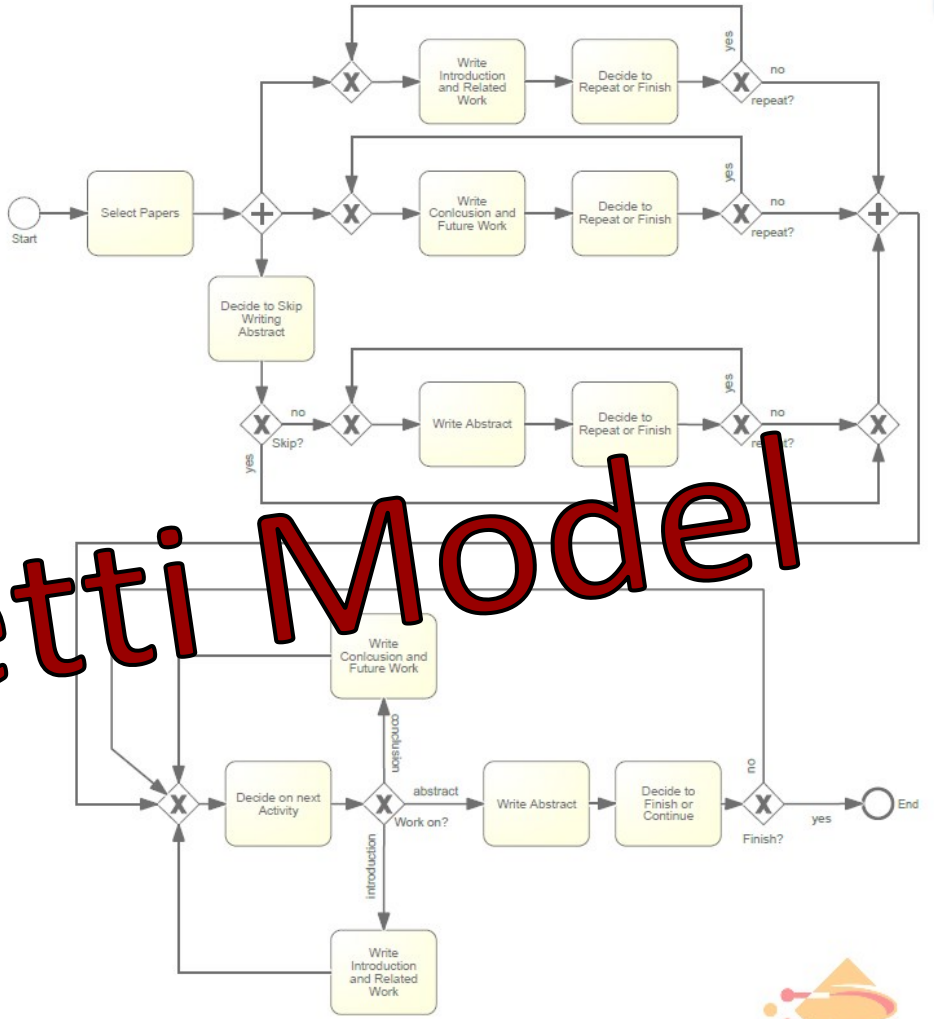
Business Process Modelling

What if we add a rule that we always need to update the abstract last?



Business Process Modelling

What if we add a rule that we always need to update the abstract last?



Spaghetti Model

eXformatics

IT University of Copenhagen

Knowledge Workers

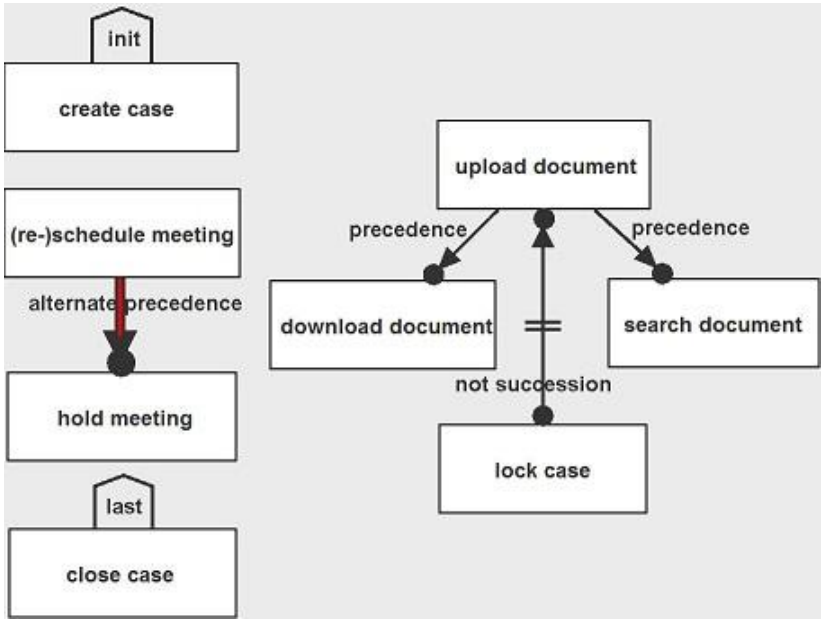
- Knowledge Workers:
  - Solve diverse problems
  - Are experts at what they do
  - Require freedom to make their own decisions
- However, rules do exist:
  - Laws
  - Business practices

Flexible Process Notations

- Flexible Process Notations:
  - Focus on describing **rules** instead of the **flow** of work
  - Offers users all possible choices that follow the rules, while still advising on best-practice
  - Are more easily adapted to change (new laws, changing business practices)

Flexible Process Notations

Not the first to suggest this: **Declare**[1]



- Declarative notation for Flexible processes
- Large set of common business constraints
- Formalized as Linear Temporal Logic

[1] M. Pesic and W.M.P. van der Aalst. A declarative approach for flexible business processes management (2006)

Flexible Process Notations

However:

- Because of the mapping from Declare to LTL to automata there is no tight coupling between design-time and run-time:
  - makes it harder to reason about execution
  - makes it harder to do run-time adaptation of declare processes
- Formal expressiveness of Declare unclear
  - At most LTL

Flexible Process Notations

Dynamic Condition Response (DCR) Graphs[1]

- New declarative notation
- Generalization of Event Structures
- Inspired by Resultmakers industrial notation
- Only four basic relations
- Strong formal expressiveness (union of regular and ω-regular languages)
- Runtime semantics based on transformation of the graph

[1] R. R. Mukkamala, T. Hildebrandt. Distributed Dynamic Condition Response Structures. (2010)

Overview

- Background
- Dynamic Condition Response (DCR) Graphs
  - Introduction
  - Hierarchy
  - Time
- DCR Graphs for Cross-organizational Processes
- Conclusion
- Demo

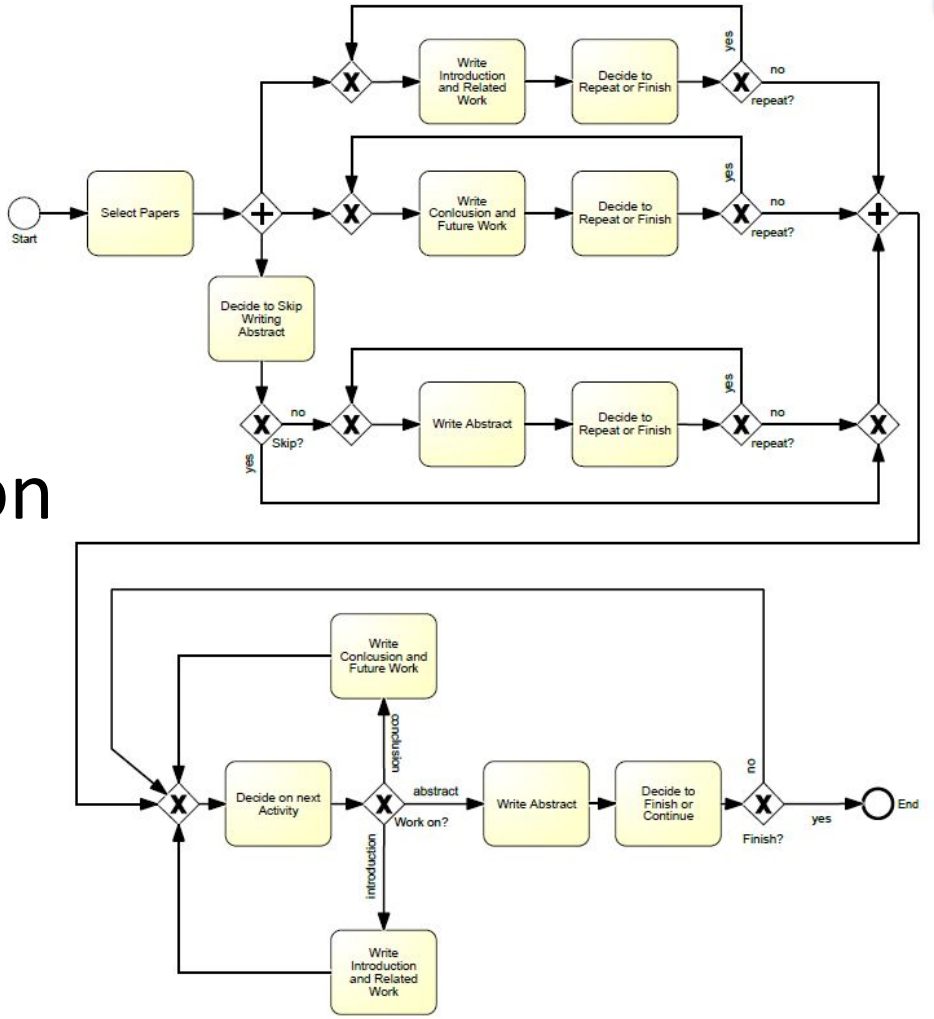
DCR Graphs

A declarative workflow notation, consisting of:

- *Events* (activities)
  - Unconstrained events can happen at any time and any number of times
- *Constraints* (rules) between events
- State represented as a *marking* consisting of *executed*, *pending* and *included* events

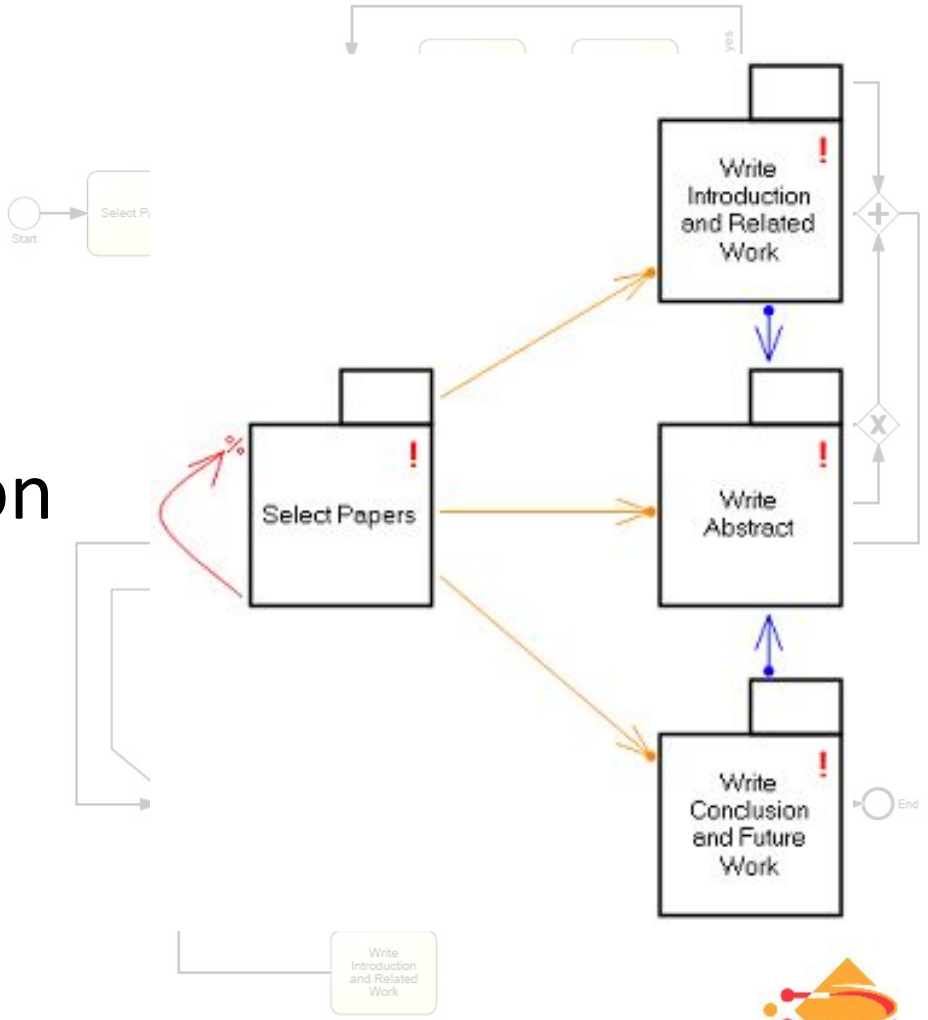
Example

- We first select papers, then:
- In any order, but at least once:
  - Write Introduction
  - Write Conclusion
  - Write Abstract
- We always update the abstract last



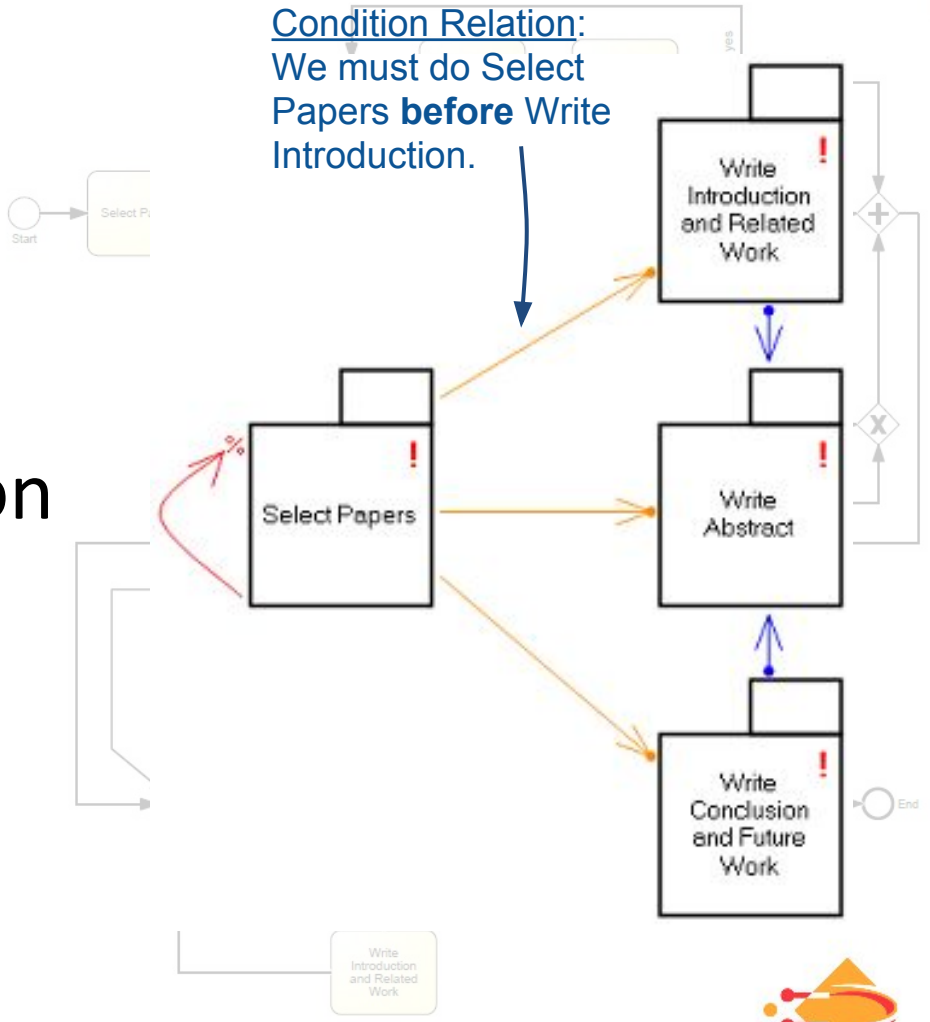
Example

- We first select papers, then:
- In any order, but at least once:
  - Write Introduction
  - Write Conclusion
  - Write Abstract
- We always update the abstract last



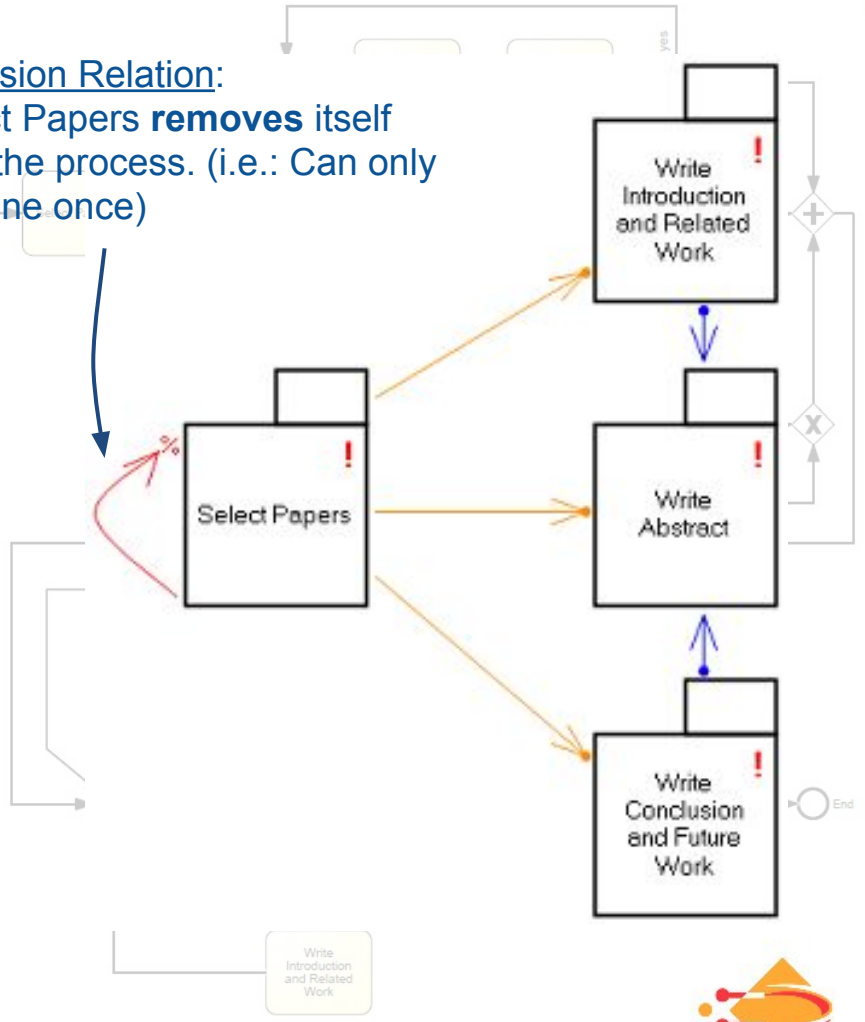
Example

- We first select papers, then:
- In any order, but at least once:
  - Write Introduction
  - Write Conclusion
  - Write Abstract
- We always update the abstract last



Example

- We first select papers, then:
- In any order, but at least once:
  - Write Introduction
  - Write Conclusion
  - Write Abstract
- We always update the abstract last

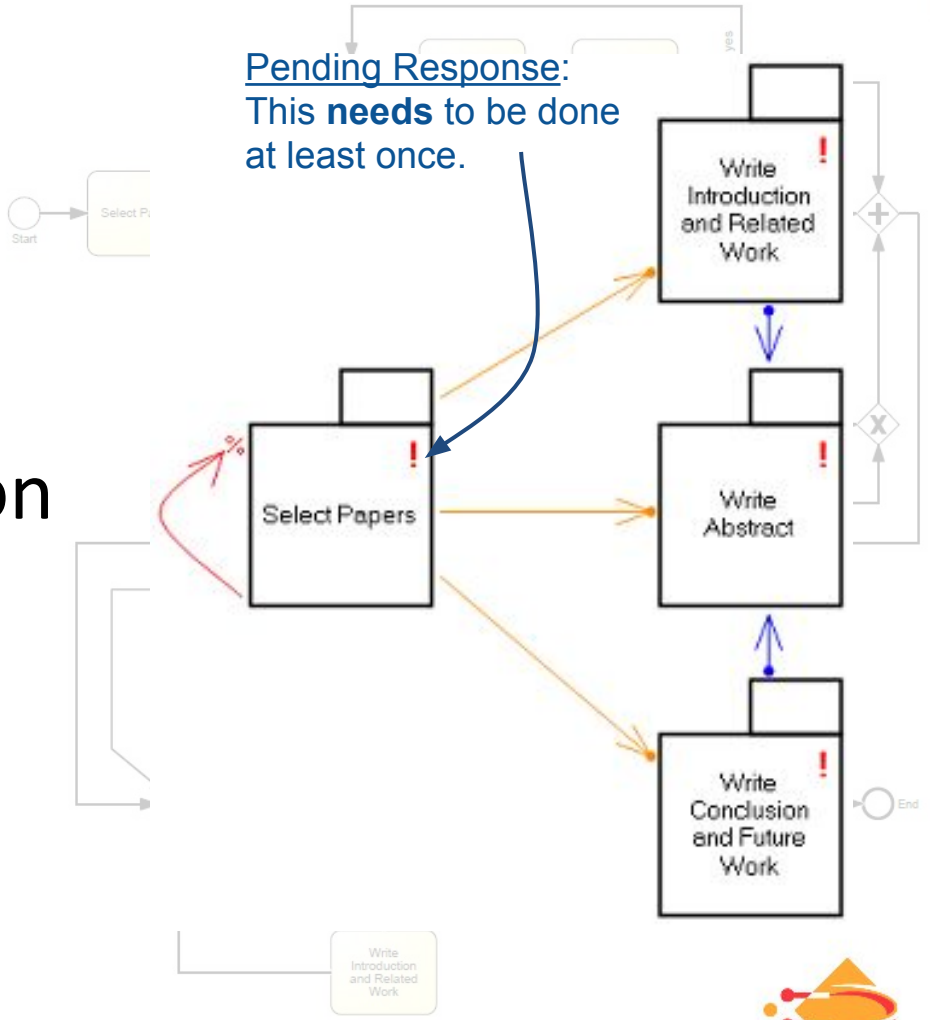


eXformatics

IT University of Copenhagen

Example

- We first select papers, then:
- In any order, but at least once:
  - Write Introduction
  - Write Conclusion
  - Write Abstract
- We always update the abstract last

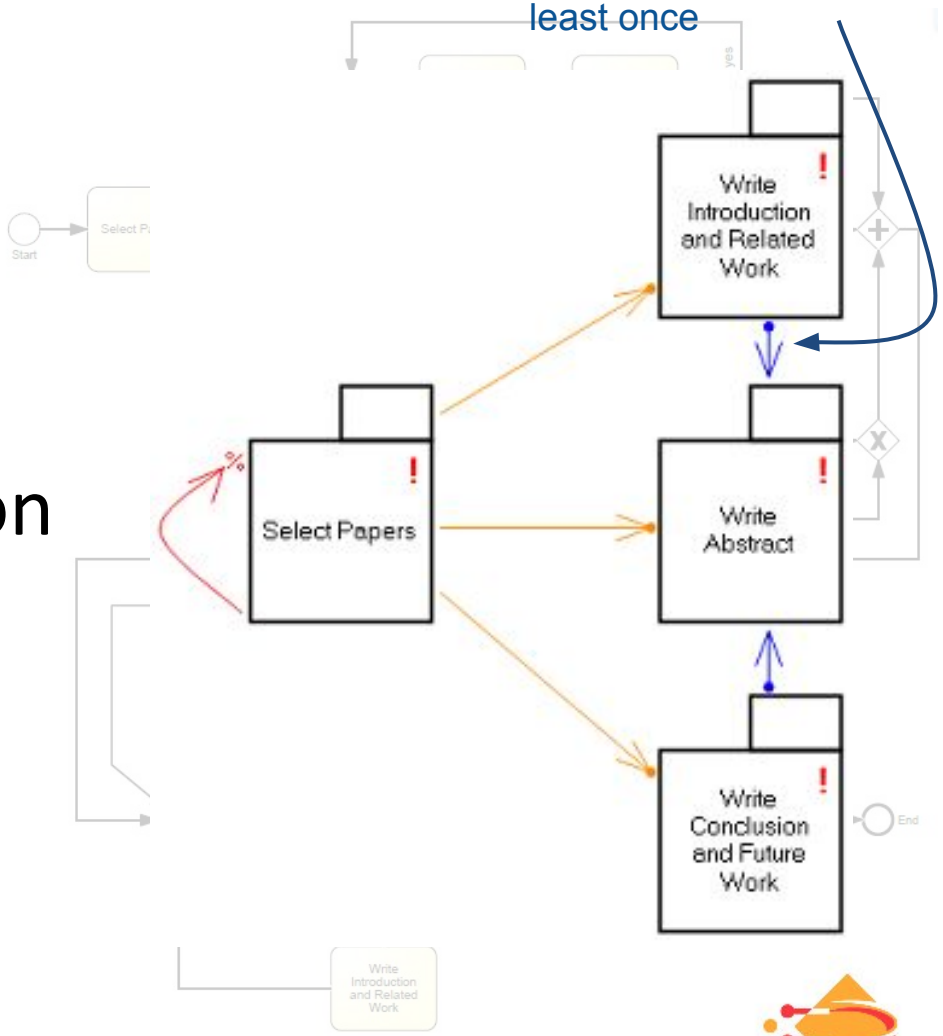


eXformatics

IT University of Copenhagen

Example

- We first select papers, then:
- In any order, but at least once:
  - Write Introduction
  - Write Conclusion
  - Write Abstract
- We always update the abstract last



Another example

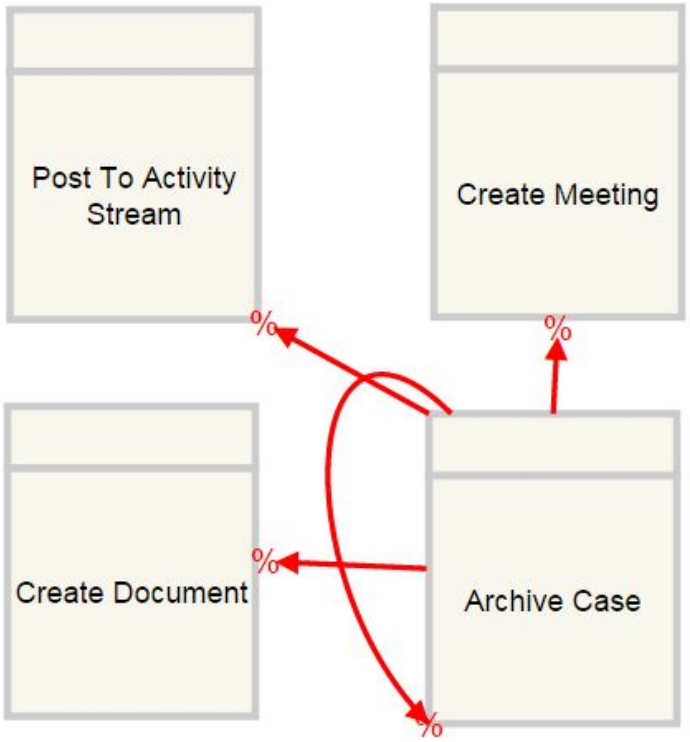
Electronic Case Management System

- Centered around concept of a case:
  - Legal cases
  - Insurance claims
  - Patient care
  - etc…
- Focuses on facilitating communication, document management and workflow

ECM Example

Three main activities:

- Post to Activity Stream
- Create Meeting
- Create Document



Archive Case closes the case by removing all activities

ECM Example

Three main activities:

- Post to Activity Stream
- Create Meeting
- Create Document

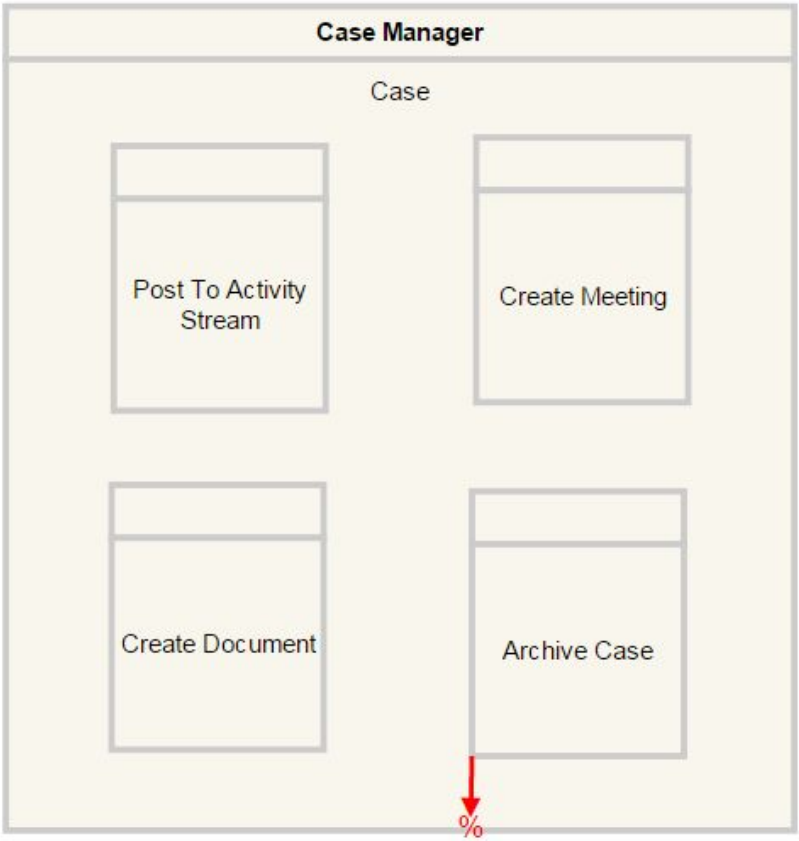
Lots of arrows…

gets a bit messy!

Archive Case closes the case by removing all activities

ECM Example - Nesting

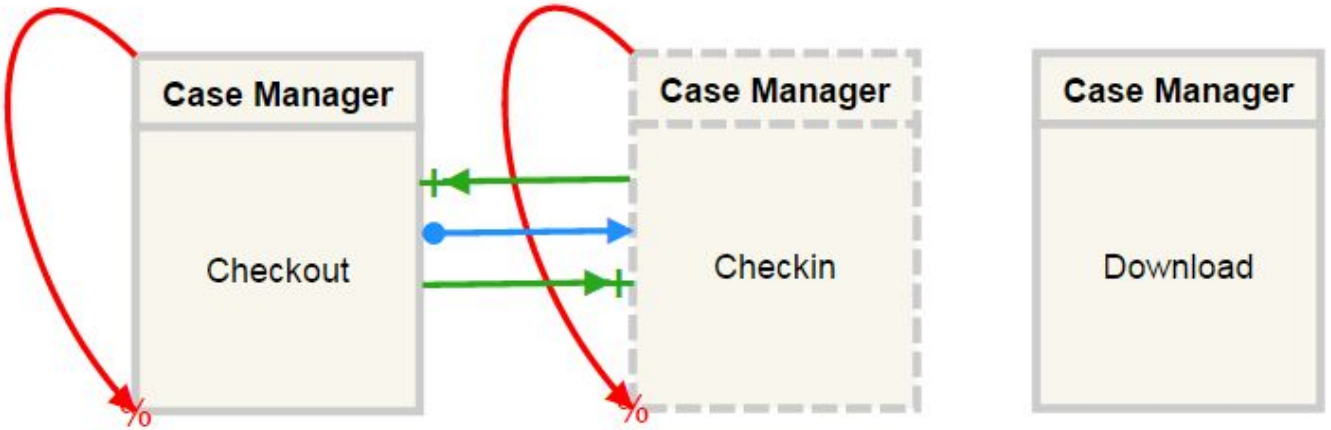
- Group activities together
- Only atomic activities are executable
- Nesting serves as a shorthand for applying relations to more than one activity



ECM Example

Document handling process

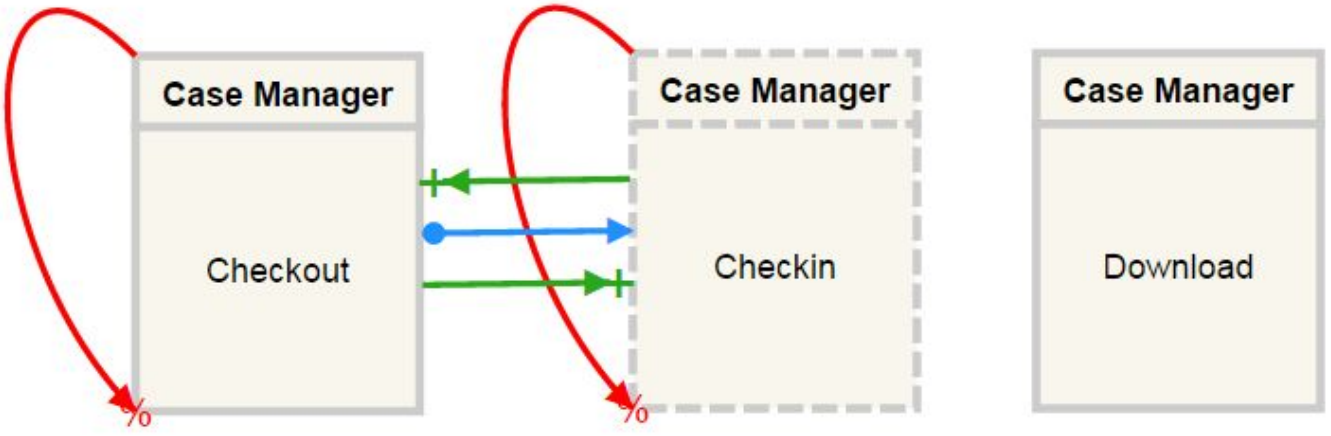
- A file is **checked in** or **checked out**
- Eventually the file should always be checked in
- A file can always be downloaded for viewing



ECM Example

Document handling process

- A file is **checked in** or **checked out**
- Eventually the file should always be checked in
- A file can always be downloaded for viewing



Initially Excluded:
When a file is created it is already checked in, so this activity is not yet enabled.

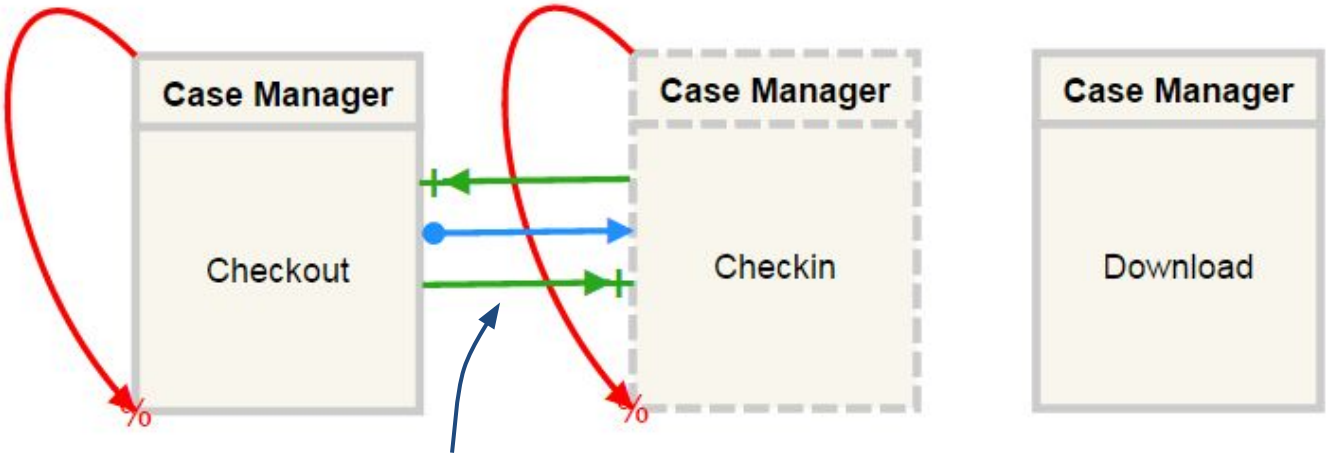
eXformatics

IT University of Copenhagen

ECM Example

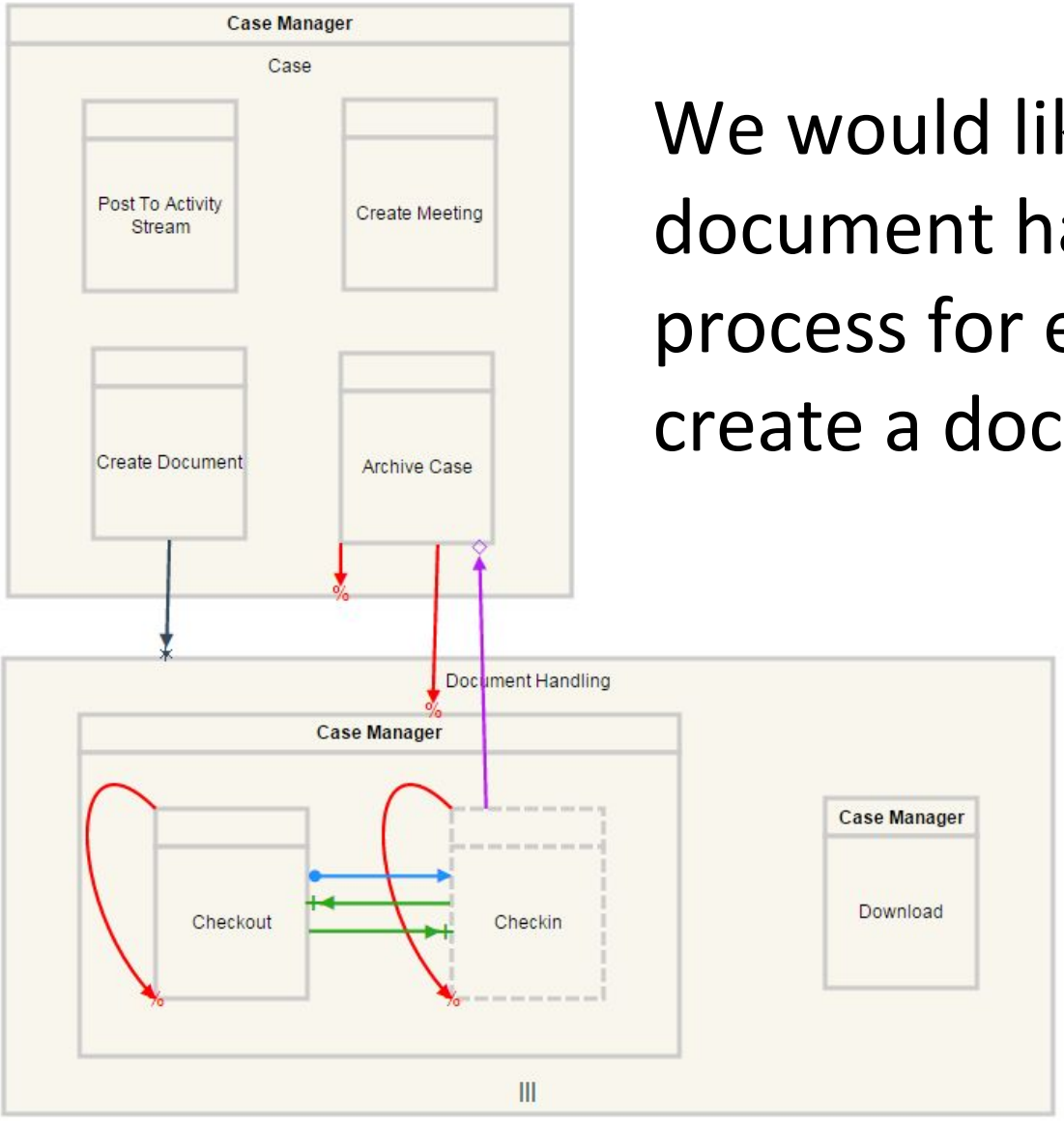
Document handling process

- A file is **checked in** or **checked out**
- Eventually the file should always be checked in
- A file can always be downloaded for viewing



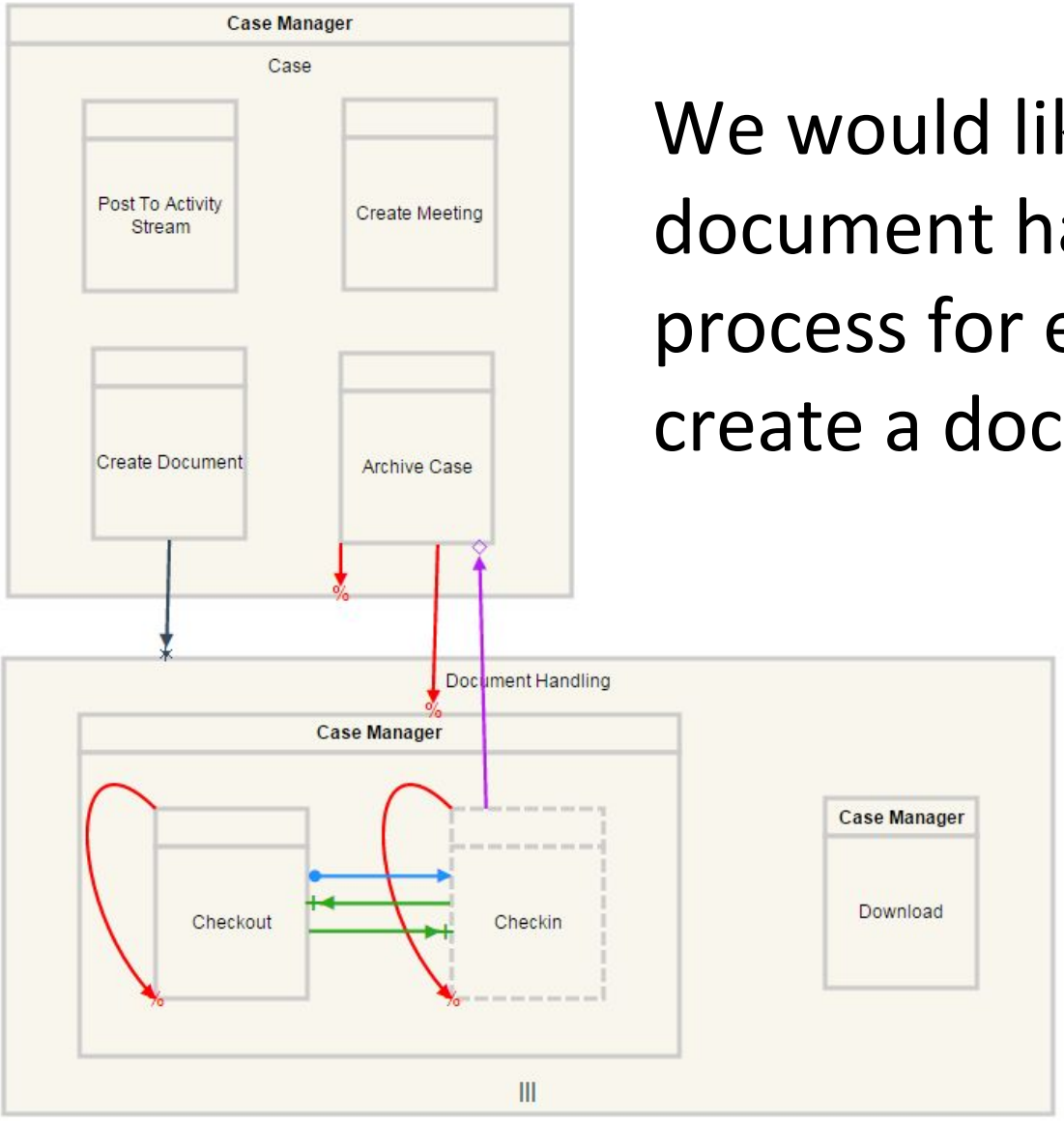
Inclusion Relation:
Checkout **Adds** Checkin (back) into the process

ECM Example - Subprocesses



We would like to start a document handling sub-process for each time we create a document.

ECM Example - Subprocesses

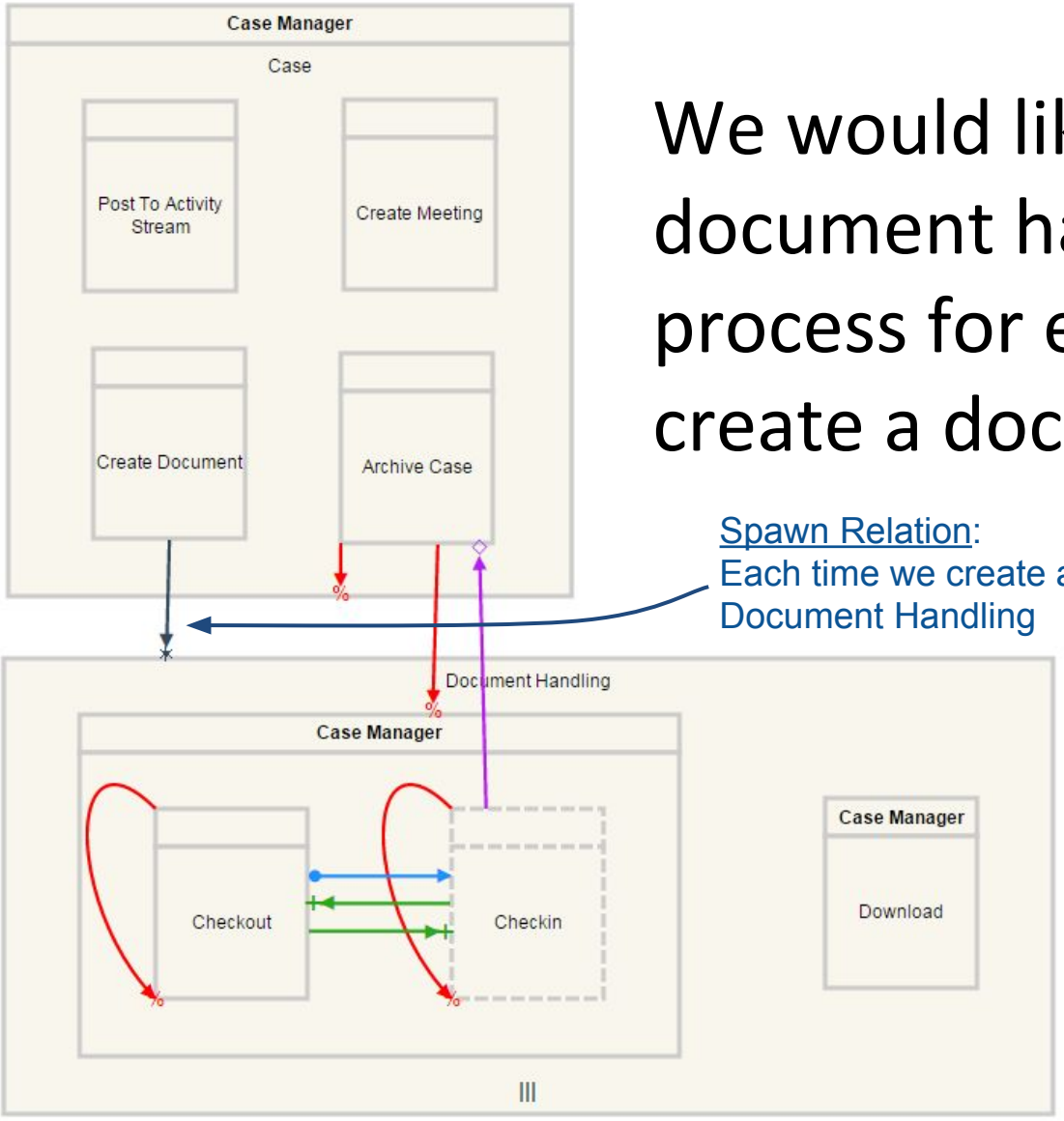


We would like to start a document handling sub-process for each time we create a document.

Multi-instance Sub-process:
A **template** of another process, does not exist on its own but needs to be **instantiated**

IT University of Copenhagen

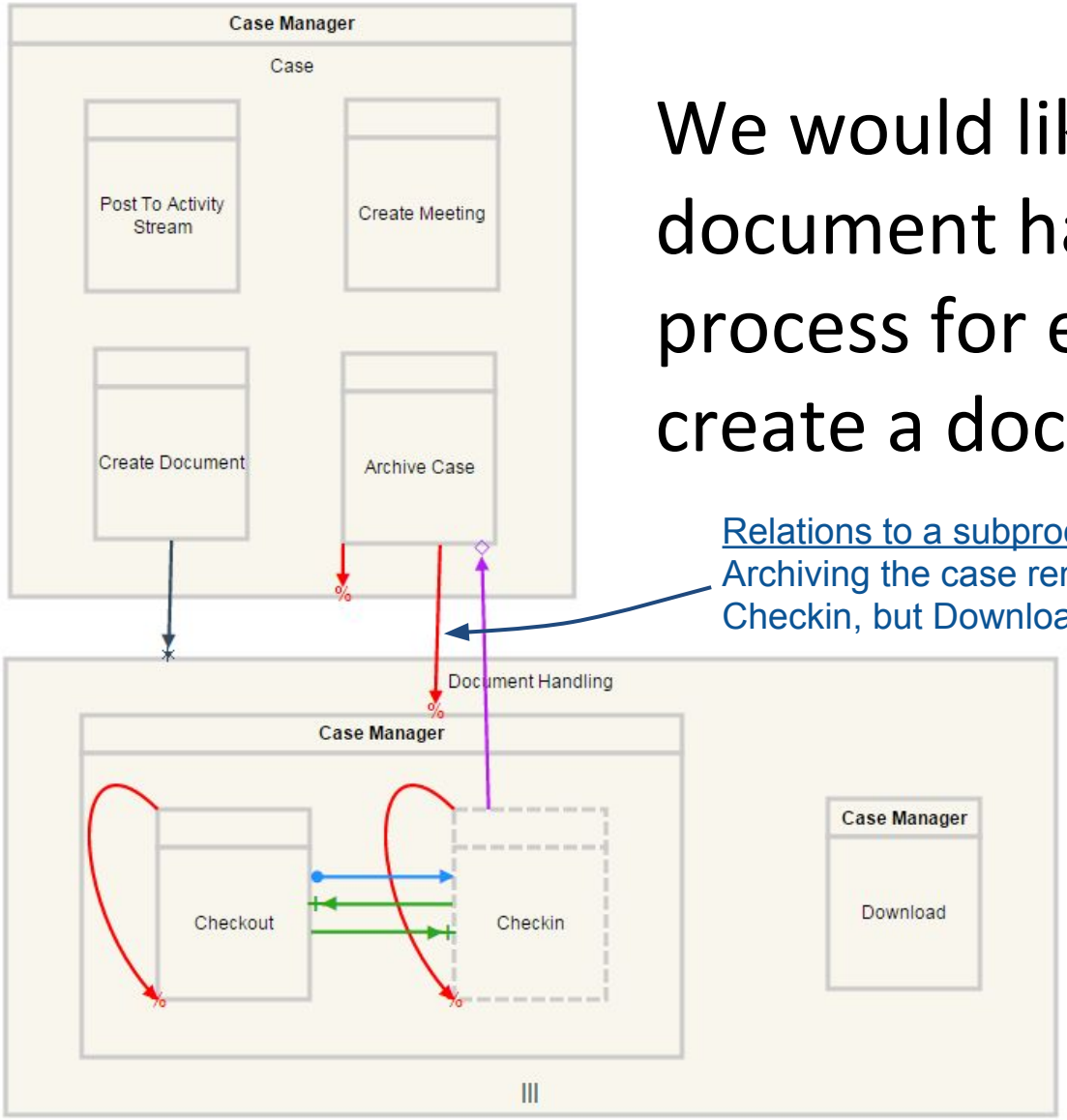
ECM Example - Subprocesses



We would like to start a document handling sub-process for each time we create a document.

Spawn Relation:
Each time we create a document, we **spawn** a new copy of Document Handling

ECM Example - Subprocesses



We would like to start a document handling sub-process for each time we create a document.

Relations to a subprocess:
Archiving the case removes **all** instances of Checkout and Checkin, but Download remains available

IT University of Copenhagen

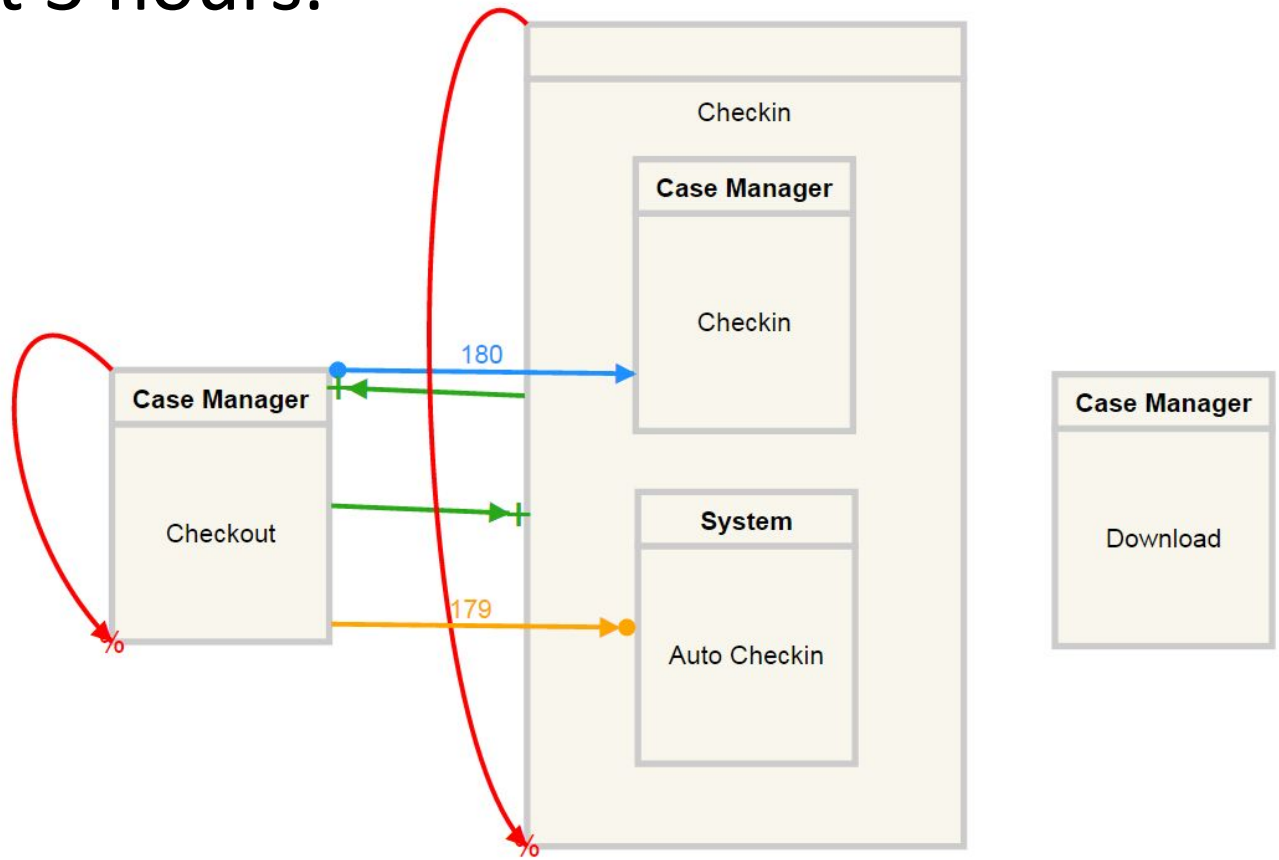
ECM Example - Subprocesses

We would like to start a document handling sub-process for each time we create a document.

Milestone Relation:
We can only archive the case while Checkin is **not pending**, i.e.: we can only archive when all document have been checked in

ECM Example - Time

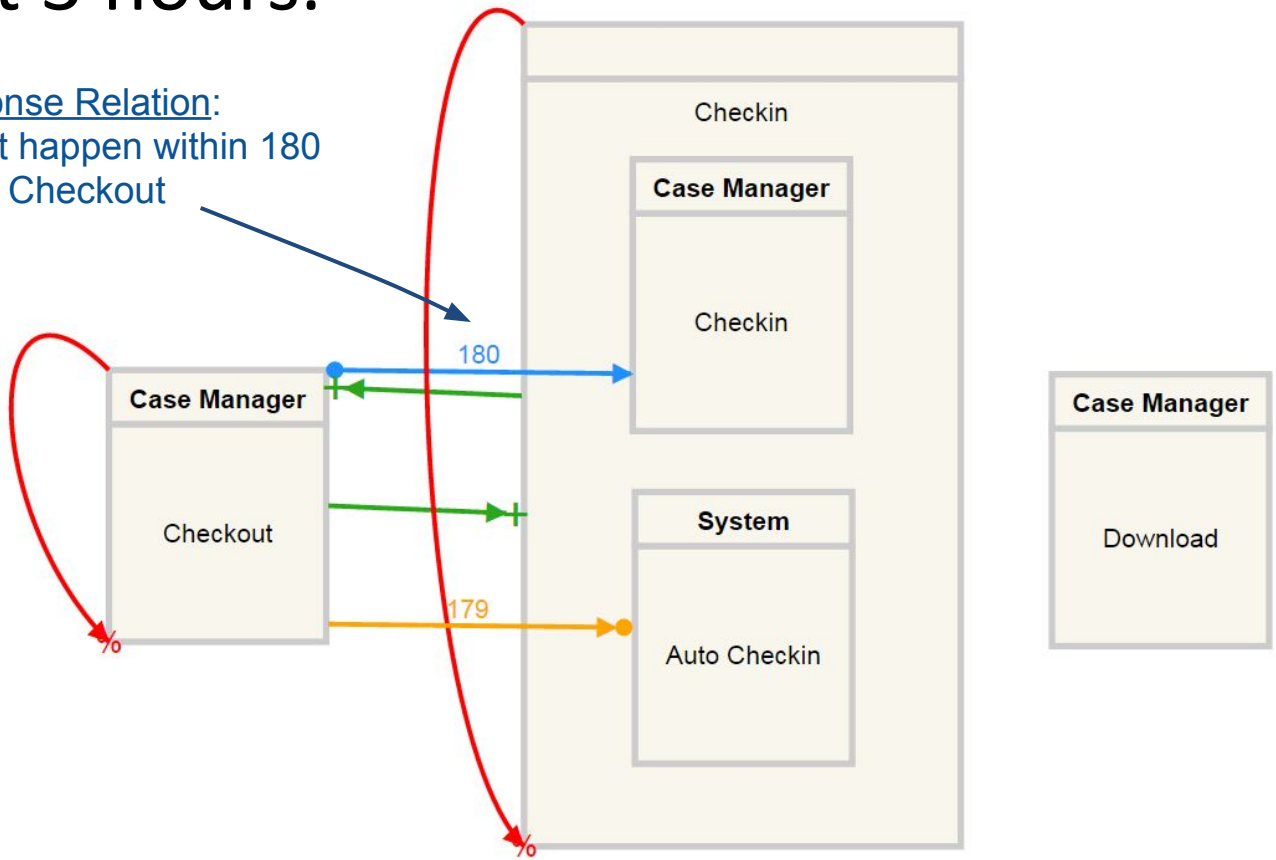
We want documents to be checked out for at most 3 hours.



ECM Example - Time

We want documents to be checked out for at most 3 hours.

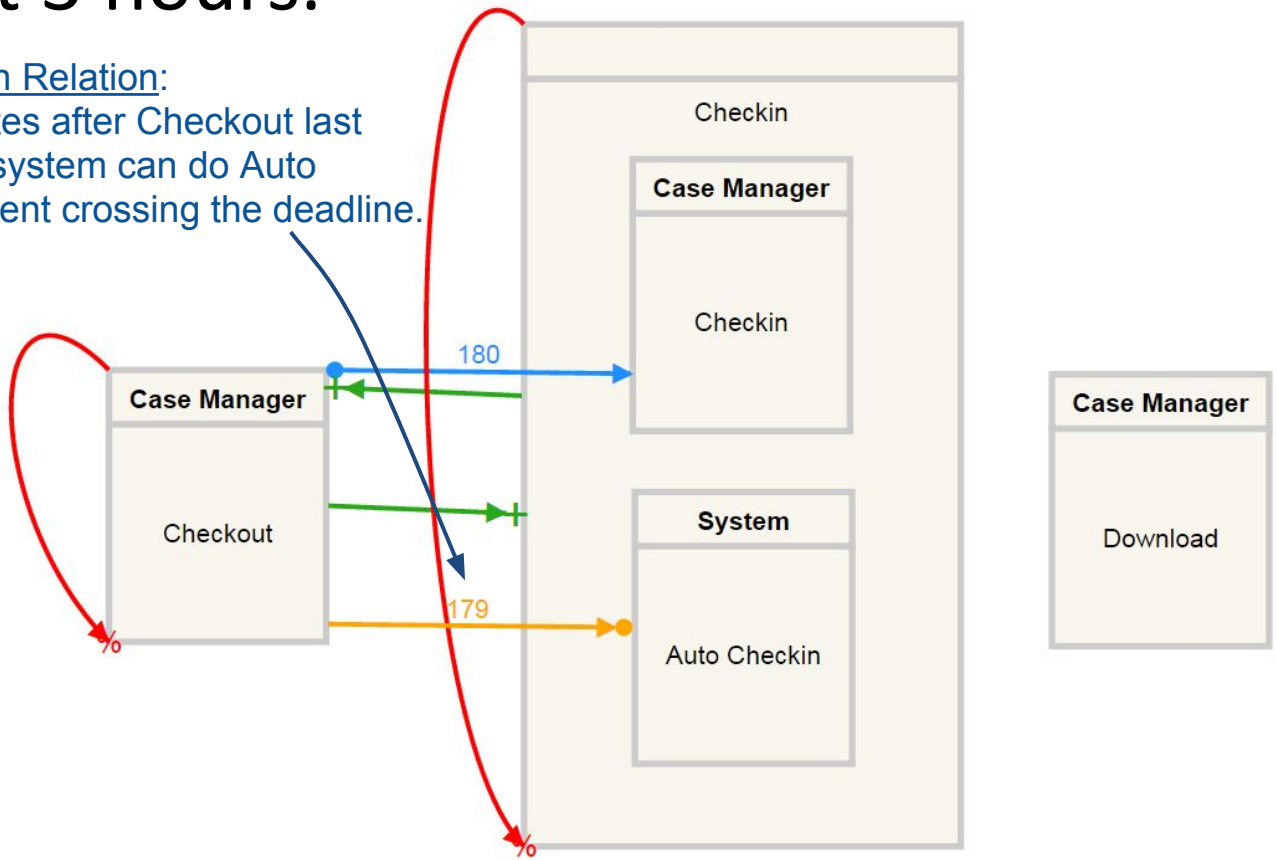
Timed Response Relation:
Checkin must happen within 180 minutes after Checkout



ECM Example - Time

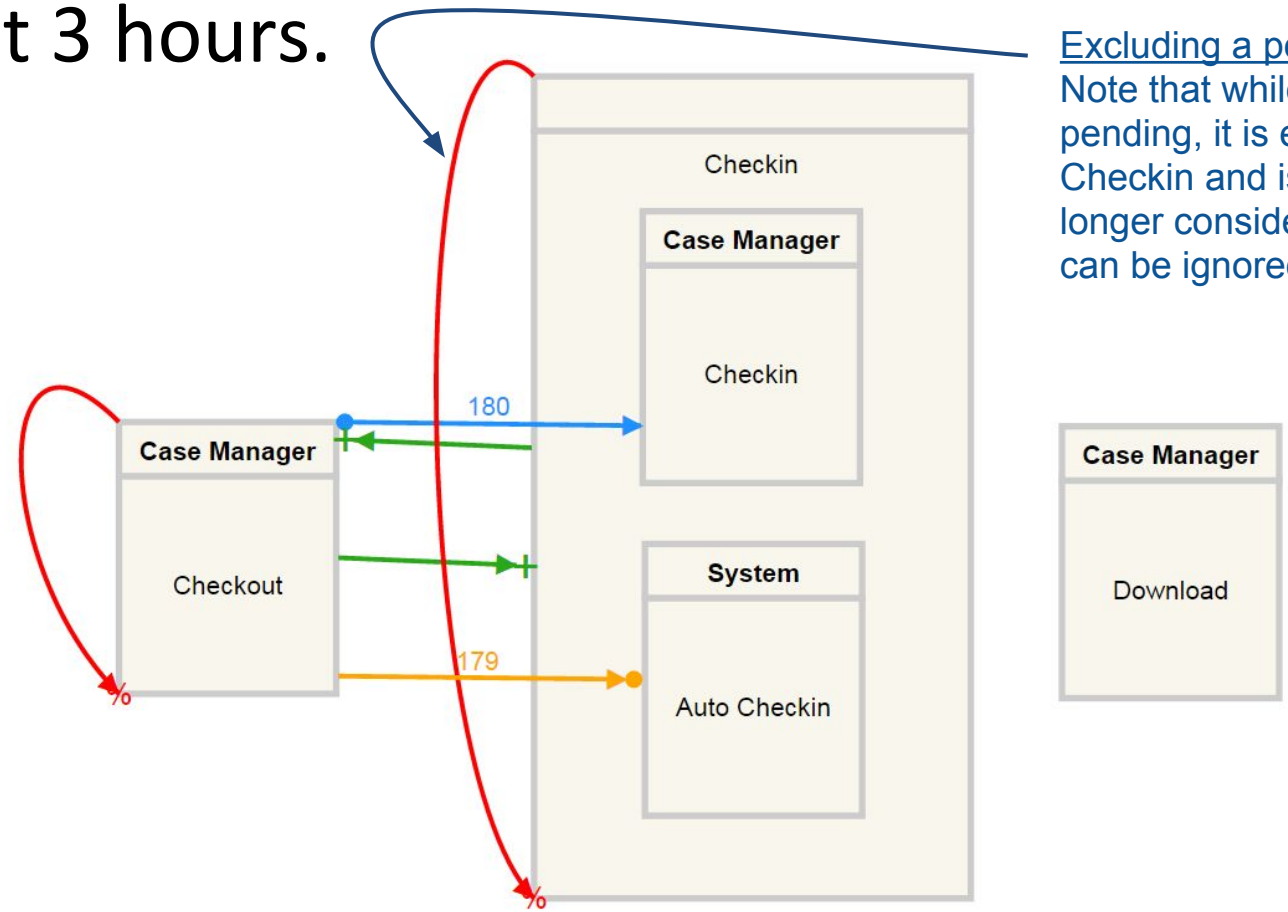
We want documents to be checked out for at most 3 hours.

Timed Condition Relation:
From 179 minutes after Checkout last happened, the system can do Auto Checkin to prevent crossing the deadline.



ECM Example - Time

We want documents to be checked out for at most 3 hours.



Excluding a pending response: Note that while Checkin is still pending, it is excluded by Auto Checkin and is therefore no longer considered relevant (and can be ignored).

Expressiveness and Verification

	Expressiveness
DCR Graphs	Union of regular and ω-regular languages
Timed DCR Graphs	Union of regular and ω-regular languages + discrete timesteps
Hi-DCR Graphs	Turing Complete

Deadlock and livelock analysis techniques exist for standard and timed DCR Graphs.[1]
Mappings exist to Büchi automata[2], (asynchronous) labelled transitions systems and Petri nets

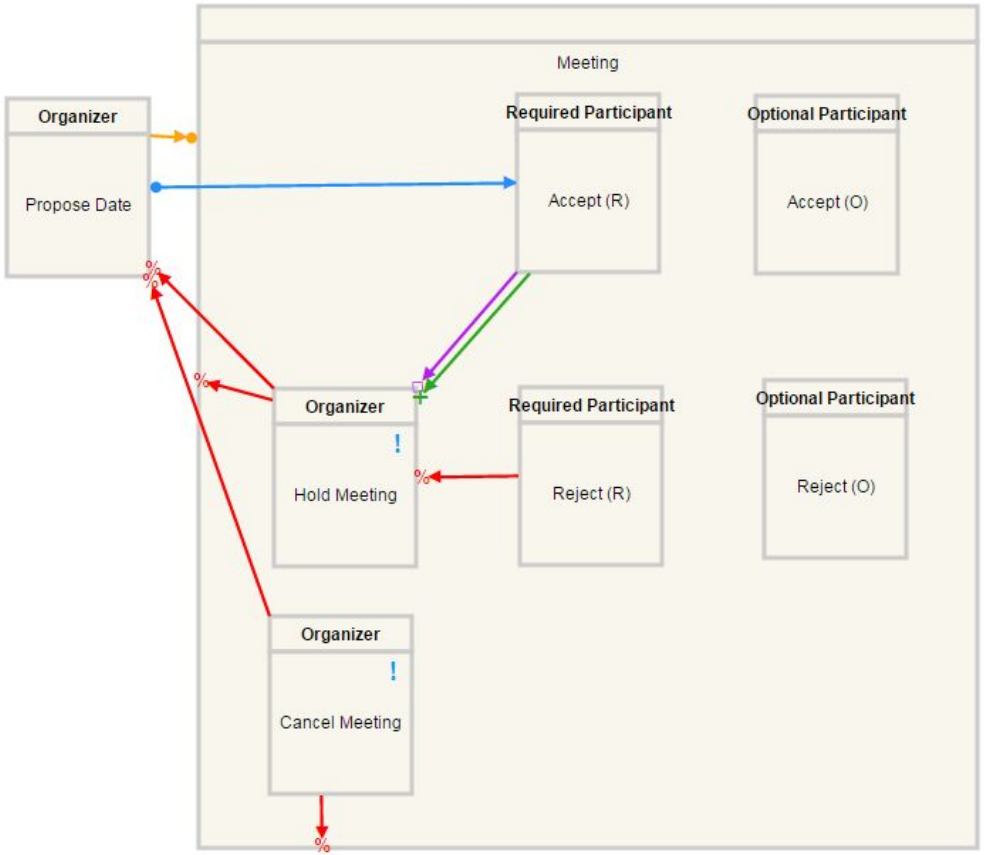
[1] R. R. Mukkamala. A Formal Model For Declarative Workflows (2012)
[2] R. R. Mukkamala and T. Hildebrandt. From Dynamic Condition Response Structures to Buchi Automata. (2010)

Overview

- Background
- Dynamic Condition Response (DCR) Graphs
- DCR Graphs for Cross-organizational Processes
  - Projection
  - Independence Relation
  - Component DCR Graphs
- Conclusion
- Demo

Projection of DCR Graphs

- Declarative notations give a great global view of the business constraints of a process
- Can we get a local view for each participant?

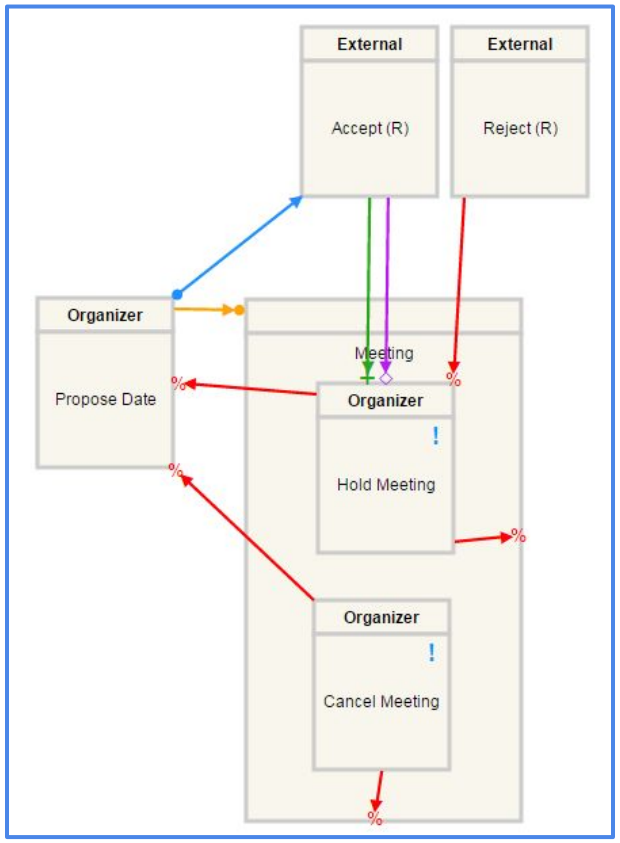


eXformatics

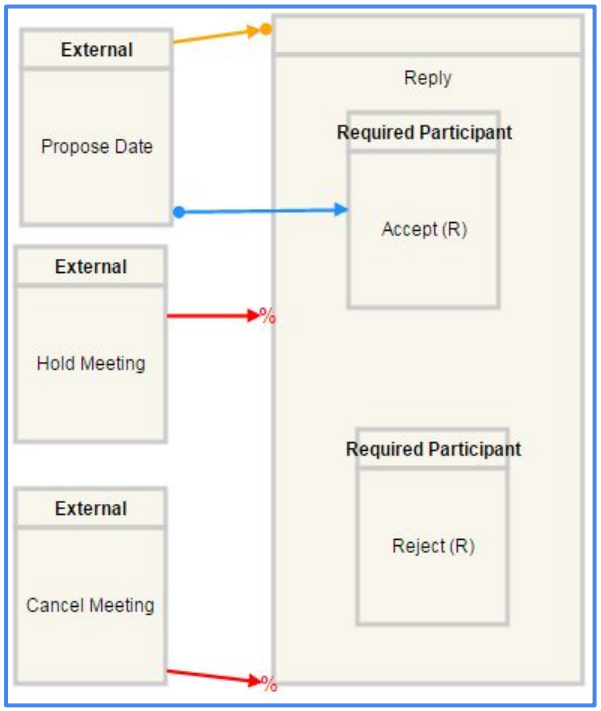
IT University of Copenhagen

Projection of DCR Graphs

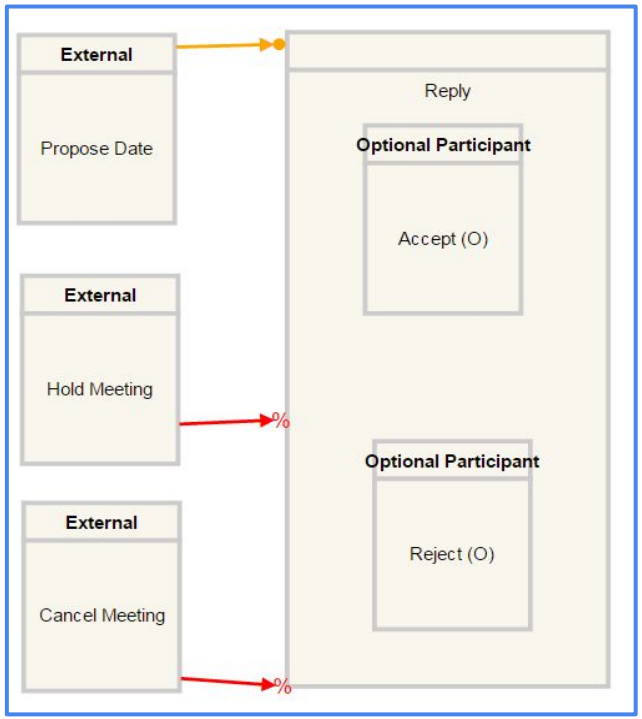
Organizers View



Required Participants View



Optional Participants View



- Each process has local events and a set of external events that it is affected by.
- It is only aware of those relations that are relevant to it.
- Only the execution of events is communicated.

eXformatics

IT University of Copenhagen

Independence Relation for DCR Graphs

Using basic projection means that communication between the distributed graphs needs to be synchronous.

If we can determine which events are *independent,* i.e. don’t affect each other's execution, we take a (partially) asynchronous approach.

An *independence relation* determines which events are independent of each other.

Independence Relation for DCR Graphs

Two events of a DCR Graph are independent if:

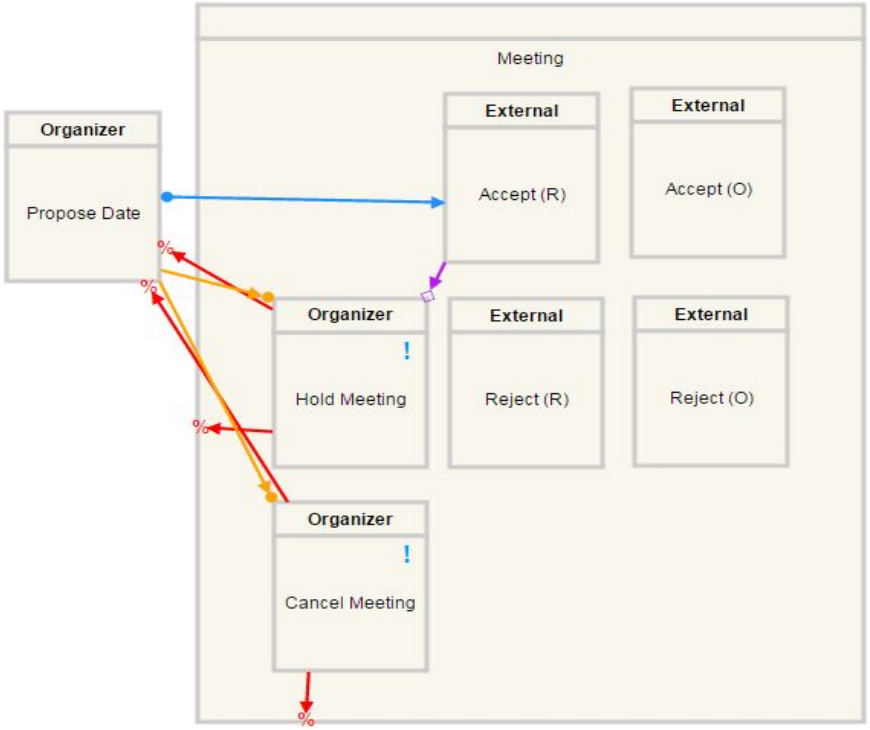
1. One does not include an event that the other excludes.
2. Neither is a response to the other.
3. Neither is a condition or milestone to the other.
4. Neither includes or excludes the other.
5. Neither includes or excludes a condition or milestone of the other.
6. Neither is a response to a milestone of the other.

Independence Relation for DCR Graphs

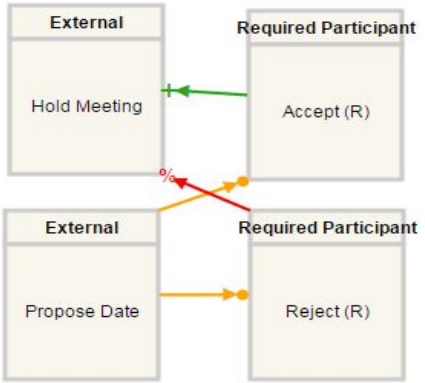
	Propose Date	Hold Meeting	Cancel Meeting	Accept (R)	Reject (R)	Accept (O)	Reject (O)
Propose Date							
Hold Meeting							
Cancel Meeting							
Accept (R)						X	X
Reject (R)						X	X
Accept (O)				X	X		X
Reject (O)				X	X	X	

Component DCR Graphs

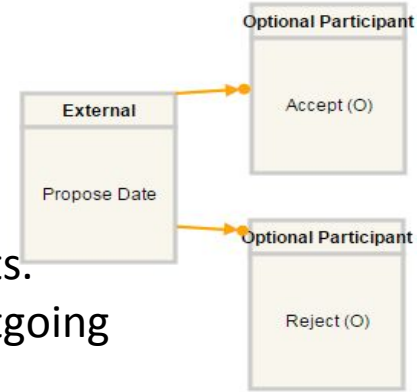
Organizers Service



Required Participants Service



Optional Participants Service



- Separate components responsible for a subset of events.
- Keeps track of incoming conditions, milestones and outgoing inclusions, exclusions and responses.
- Updates state of external event at the responsible component.
- Before making state changes all affected events are locked.

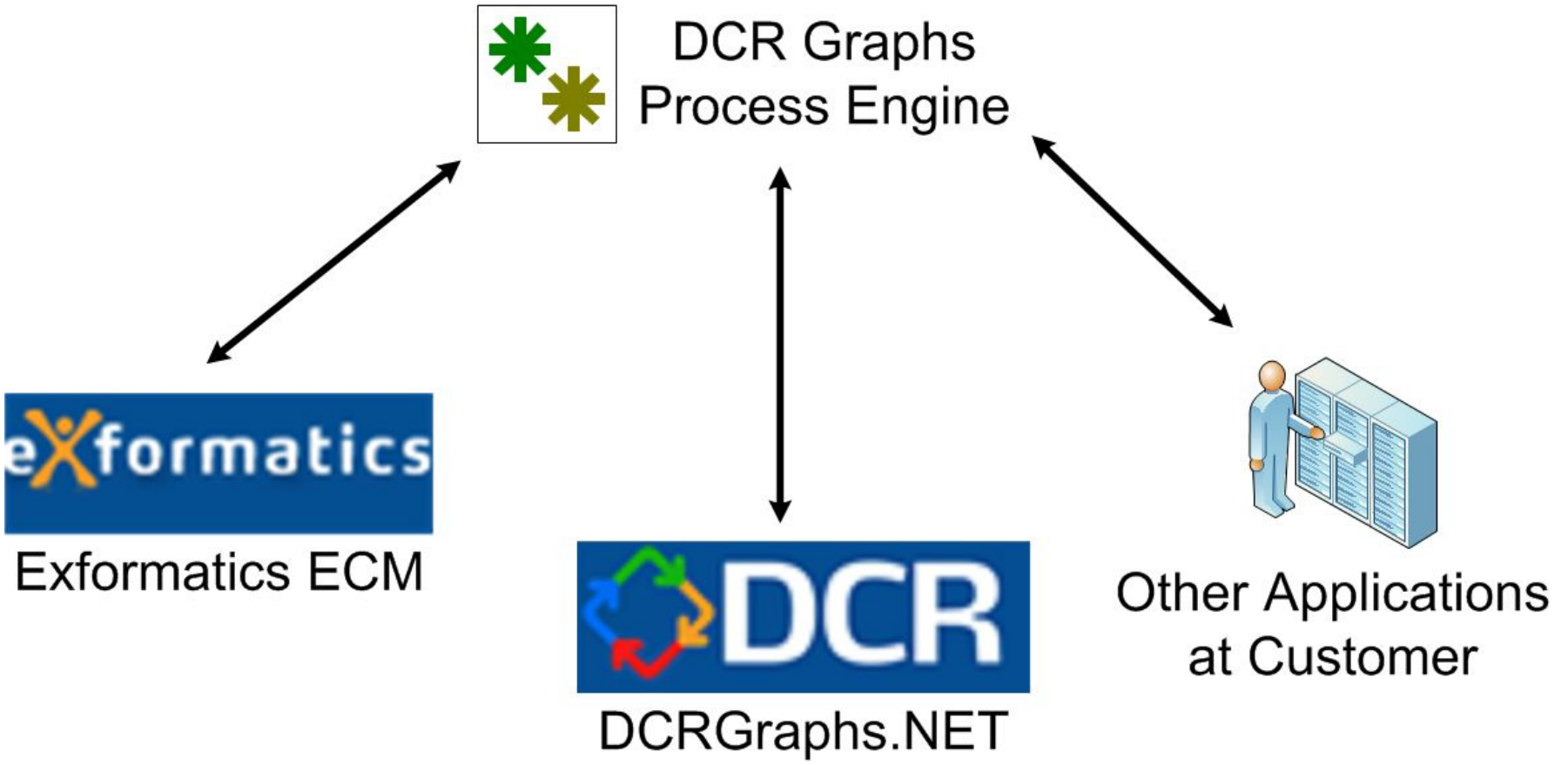
eXformatics

IT University of Copenhagen

Overview

- Background
- Dynamic Condition Response (DCR) Graphs
- DCR Graphs for Cross-organizational Processes
- Conclusion
  - Industrial Results
  - Conclusion
- Demo

Exformatics DCR Graphs Tools



Conclusion

In this talk we showed:

- How we can model flexible knowledge-centered processes with declarative process notations.
- Dynamic Condition Response Graphs, such a declarative process notation.
- Extensions to DCR Graphs driven by business needs
- Techniques for the distribution of declaratively modelled processes

Demo

http://www.dcrgraphs.net/

Questions

Thank you for listening!