

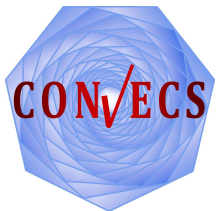
# Checking Business Process Evolution

Gwen Salaün

Université Grenoble Alpes, LIG, Inria, France

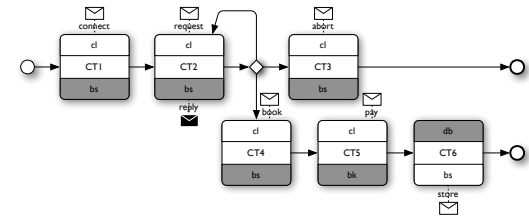
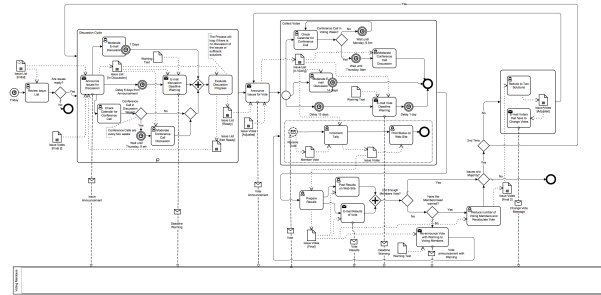
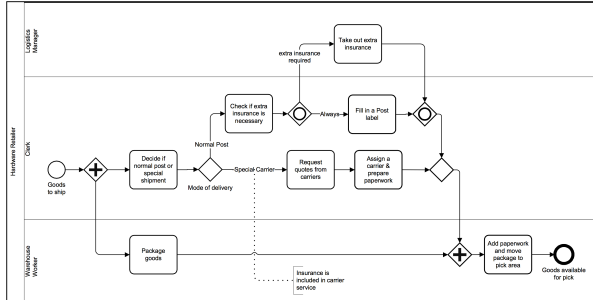
joint work with

Ajay Krishna and Pascal Poizat



# Introduction (1/2)

- A business process is a set of structured, related activities or tasks designed to produce a specific output for a customer or market
- BPMN 2.0 (Business Process Modelling Notation) was published as an ISO/IEC standard in 2013



- Modern software exhibits a high degree of dynamicity and is subject to continuous evolution

# Introduction (2/2)

- Given two BPMN business processes, we want to support the process designer in the **evolution activity** with **automated verification** techniques
- **Formal modelling** and **analysis** is important and required to ensure correctness, efficiency, and quality of the whole process execution
- Our contributions:
  - An **LTS** (Labelled Transition System) **semantics** for a subset of BPMN obtained via **process algebra encoding**
  - A set of **evolution notions** based on LTS equivalences / preorders that one can use to compare two processes
  - A **tool support** for fully automating the evolution checks that can be accessed via a **Web application**

# Outline

1. BPMN
2. From BPMN to LTS
3. Process Comparison
4. Tool Support
5. Concluding Remarks

# BPMN

## BPMN 2.0 - Business Process Model and Notation

<http://bpmb.de/poster>

### Activities

- Task**: A Task is a unit of work, like a job to be performed. This is the most basic type of activity in BPMN. It is used to model a task in a process.
- Sub-process**: A Sub-process is a unit of work, like a job to be performed, that is used to model a task in a process. It is used to model a task in a process that is used to model a task in a process.
- Event-based Sub-process**: An Event-based Sub-process is a unit of work, like a job to be performed, that is used to model a task in a process. It is used to model a task in a process that is used to model a task in a process.
- Call Activity**: A Call Activity is a unit of work, like a job to be performed, that is used to model a task in a process. It is used to model a task in a process that is used to model a task in a process.

**Activity Markers**

- Task Process Marker
- Loop Marker
- Parallel Marker
- Sequence Marker
- Event-based Marker
- Complex Marker

**Task Types**

- Start Task
- Intermediate Task
- End Task
- Start Task
- Intermediate Task
- End Task

**Dependencies**

- Start Task
- Intermediate Task
- End Task

### Conversations

**Conversation Diagram**

**Conversation Diagram**

Participants: Participant A, Participant B, Participant C.

Interactions: Participant A sends a message to Participant B, Participant B sends a message to Participant C, Participant C sends a message to Participant A.

### Choreographies

**Choreography Diagram**

**Choreography Diagram**

Participants: Participant A, Participant B, Participant C.

Interactions: Participant A sends a message to Participant B, Participant B sends a message to Participant C, Participant C sends a message to Participant A.

### Events

	Start	Intermediate	End
Start	Start Event		
Intermediate		Intermediate Event	
End			End Event

**Event Types**

- Start Event
- Intermediate Event
- End Event

### Gateways

- Exclusive Gateway**: When a task is completed, only one of the outgoing flows is taken.
- Parallel Gateway**: All outgoing flows are taken simultaneously.
- Inclusive Gateway**: At least one of the outgoing flows is taken.
- Complex Gateway**: Any combination of outgoing flows is taken.
- Event-based Gateway**: The outgoing flow is determined by an event.
- OR Gateway**: At least one of the outgoing flows is taken.
- XOR Gateway**: Exactly one of the outgoing flows is taken.
- AND Gateway**: All outgoing flows are taken simultaneously.
- OR-AND Gateway**: At least one of the outgoing flows is taken, and all outgoing flows are taken simultaneously.
- OR-OR Gateway**: At least one of the outgoing flows is taken, and all outgoing flows are taken simultaneously.
- OR-AND Gateway**: At least one of the outgoing flows is taken, and all outgoing flows are taken simultaneously.

### Collaboration Diagram

**Collaboration Diagram**

Participants: Participant A, Participant B, Participant C.

Interactions: Participant A sends a message to Participant B, Participant B sends a message to Participant C, Participant C sends a message to Participant A.

### Swimlanes

**Swimlane Diagram**

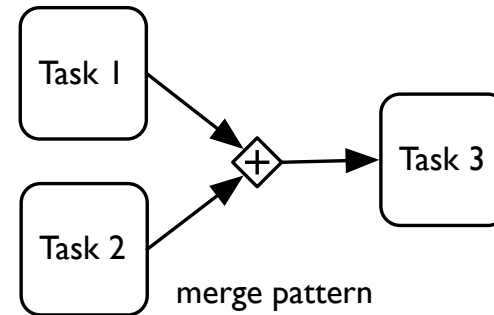
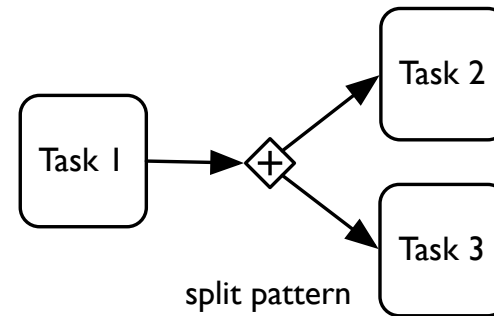
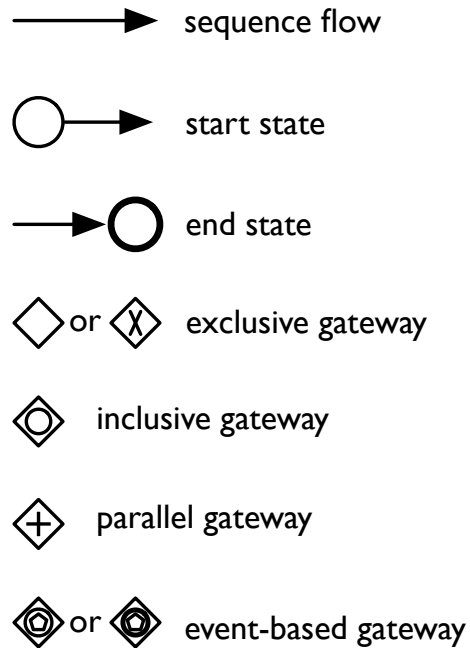
Participants: Participant A, Participant B, Participant C.

Interactions: Participant A sends a message to Participant B, Participant B sends a message to Participant C, Participant C sends a message to Participant A.

### Data

- Data Object**: A Data Object is a unit of data that is used to model a task in a process.
- Data Store**: A Data Store is a unit of data that is used to model a task in a process.
- Data Collection**: A Data Collection is a unit of data that is used to model a task in a process.
- Data Store**: A Data Store is a unit of data that is used to model a task in a process.
- Message**: A Message is a unit of data that is used to model a task in a process.

# Control Flows and Gateways



# Semantics

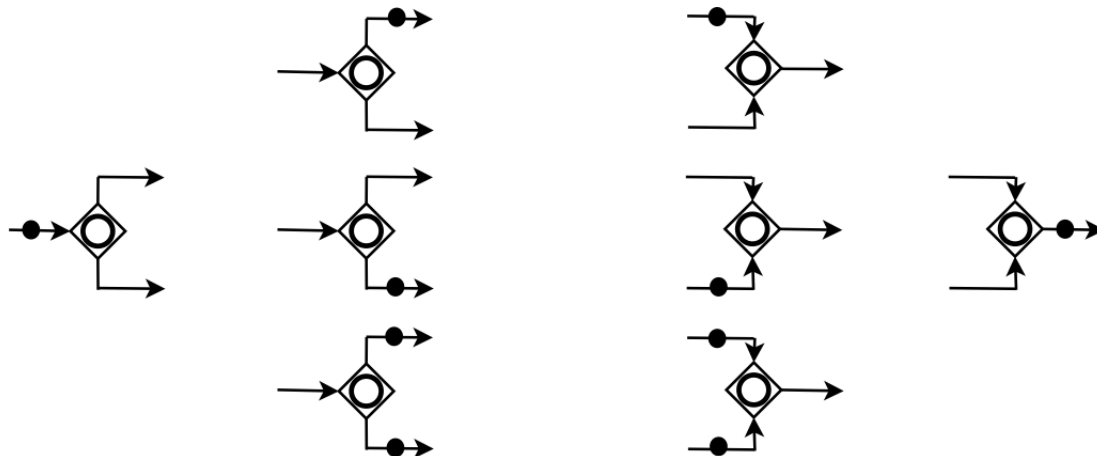
Exclusive gateway (similar for Event-based gateway): split (left) and merge (right)



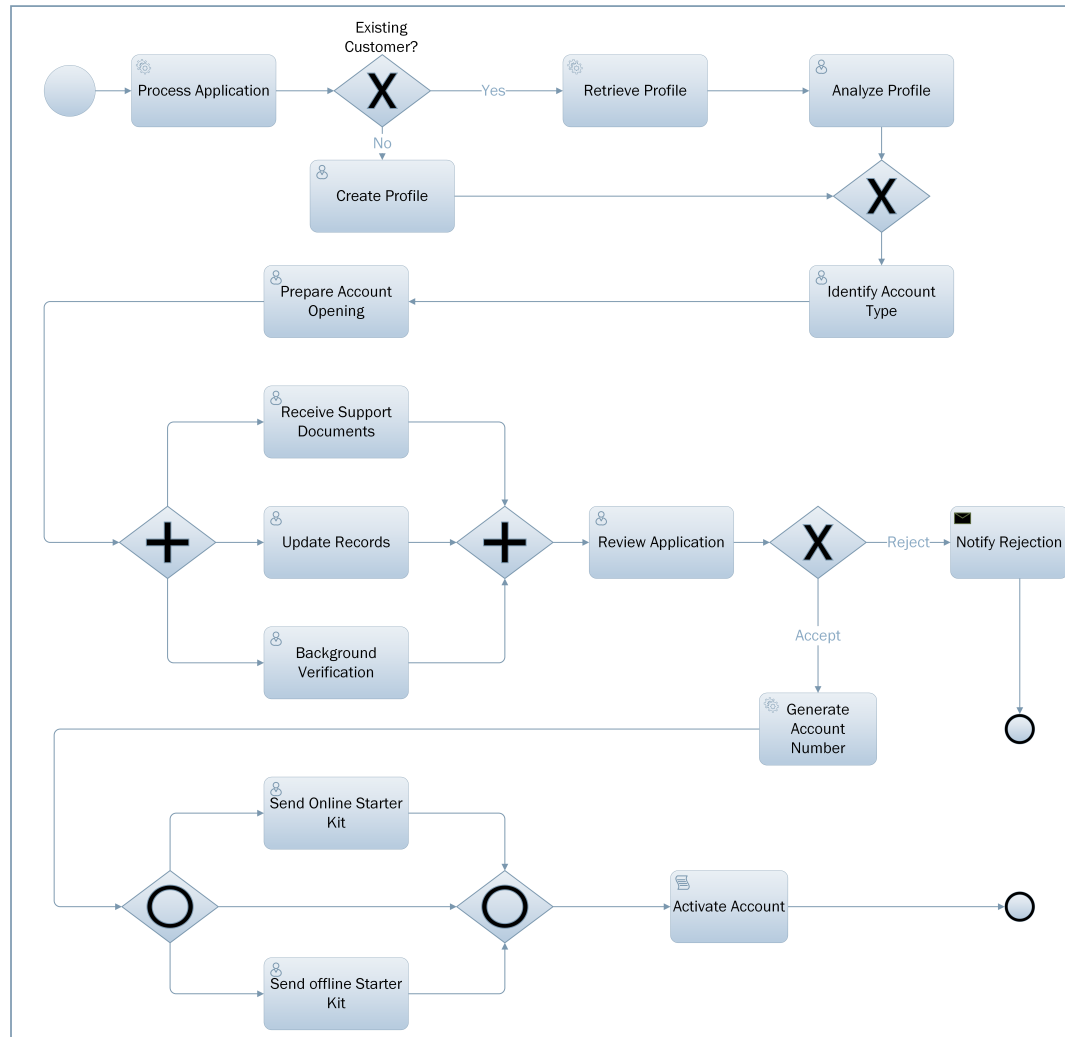
Parallel gateway : split (left) and merge (right)



Inclusive gateway : split (left) and merge (right)



# Example of BPMN Process



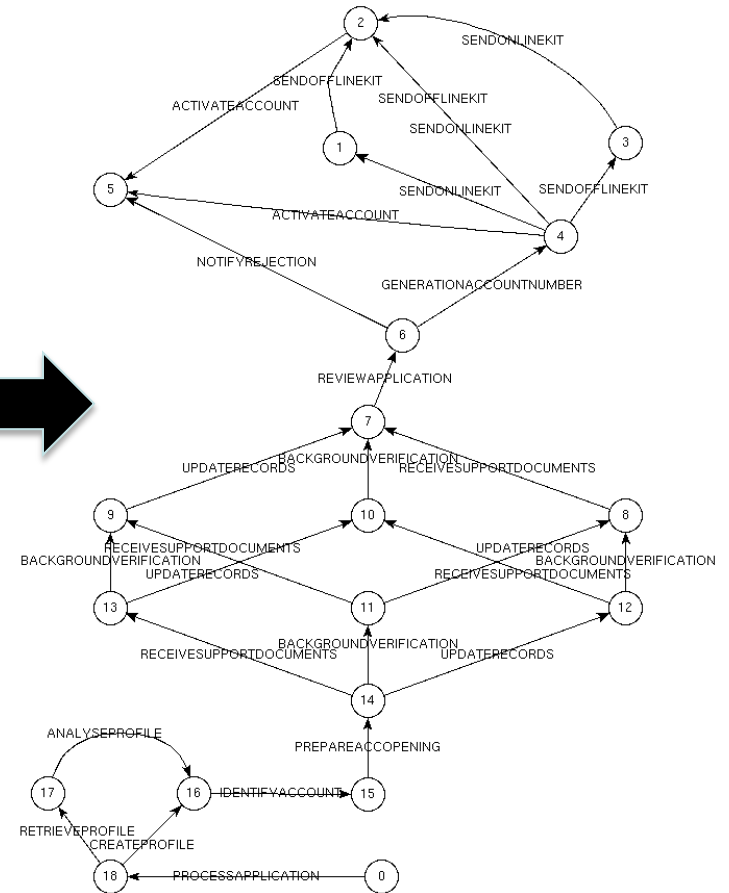
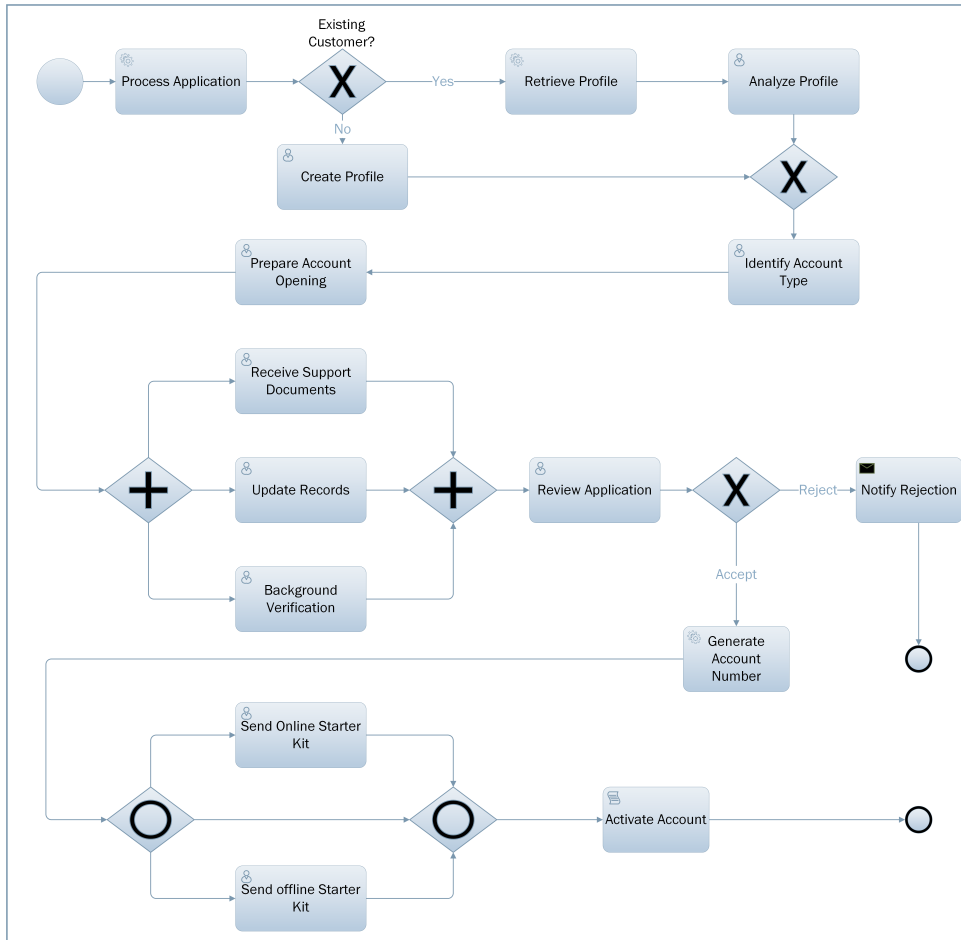
Bank account opening process



# Outline

1. BPMN
- 2. From BPMN to LTS**
3. Process Comparison
4. Tool Support
5. Concluding Remarks

# LTS Models



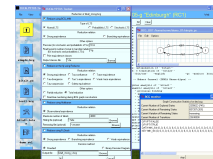
# LNT

- LOTOS NT (LNT) is a **value-passing process algebra** with user-friendly syntax and operational semantics
- LNT is an **imperative-like language** where you can specify **data types**, **functions** (pattern matching and recursion), and **processes**

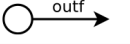
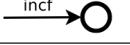
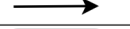
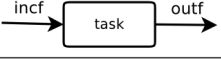
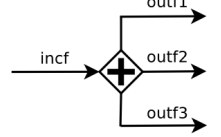
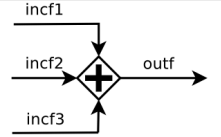
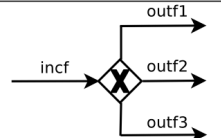
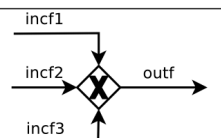
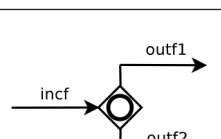
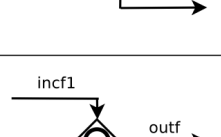
- Excerpt of the **LNT process grammar**:

```
B ::=      stop      | G(!E, ?X) where E'   | if E then B1 else B2 end if  
      |      x:=E      | hide G in B end hide | P [G1,...,Gm] (E1,...,En)  
      |      select B1 [] ... [] Bn end select | B1 ; B2  
      |      par G in B1 || ... || Bn end par
```

- **Compilation to LTS** and **verification** using the CADP toolbox



# LNT Encoding (1/2)

BPMN construct	BPMN notation	LNT encoding
Initial event		<code>begin ; outf</code>
End event		<code>incf ; finish</code>
Sequence flow		<code>loop begin ; finish end loop</code>
Task		<code>loop incf ; task ; outf end loop</code>
Parallel gateway (split)		<code>incf ; par   outf1    outf2    outf3 end par</code>
Parallel gateway (merge)		<code>par   incf1    incf2    incf3 end par ; outf</code>
Exclusive gateway (split)		<code>incf ; select   outf1 [] outf2 [] outf3 end select</code>
Exclusive gateway (merge)		<code>select   incf1 [] incf2 [] incf3 end select ; outf</code>
Inclusive gateway (split)		<code>incf ; select (* si if one matching merge *)   outf1 ; s1   [] outf2 ; s2   [] par outf1    outf2 end par ; s3 end select</code>
Inclusive gateway (merge)		<code>select (* si if one matching split *)   s1 ; incf1   [] s2 ; incf2   [] s3 ; par incf1    incf2 end par end select ; outf</code>

# LNT Encoding (2/2)

```
process main [processApplication:any, reply:any, createProfile:any, ...] is
  hide begin:any, finish:any, flow1_begin:any, flow1_finish:any, ... in
    par flow1_begin, flow1_finish, flow2_begin, flow2_finish, ... in
      par
        flow [flow1_begin, flow1_finish] || ... || flow [flow29_begin, flow29_finish]
      end par
    ||
      par
        init [begin,flow1_begin]
        || final [flow21_finish, finish] || final [flow27_finish, finish]
        || task [flow1_finish, processApplication, flow2_begin] || task [...] || ...
        || xorsplit [flow2_finish, flow3_begin, flow4_begin]
        || xormerge [flow6_finish, flow7_finish, flow29_begin]
      end par
    end par
  end hide
end process
```

# Outline

1. BPMN
2. From BPMN to LTS
- 3. Process Comparison**
4. Tool Support
5. Concluding Remarks

# Evolution

Given two BPMN processes  $P_1$  and  $P_2$ , we can define several notions of comparison using concurrency theory

- Conservative evolution: both processes exhibit exactly the same behaviour

$$\text{LTS}(P_1) =_{\text{br}} \text{LTS}(P_2)$$

- Inclusive / exclusive evolution: one process simulated by the other

$$\text{LTS}(P_1) <_{\text{br}} \text{LTS}(P_2) \quad (\text{LTS}(P_1) >_{\text{br}} \text{LTS}(P_2), \text{ resp.})$$

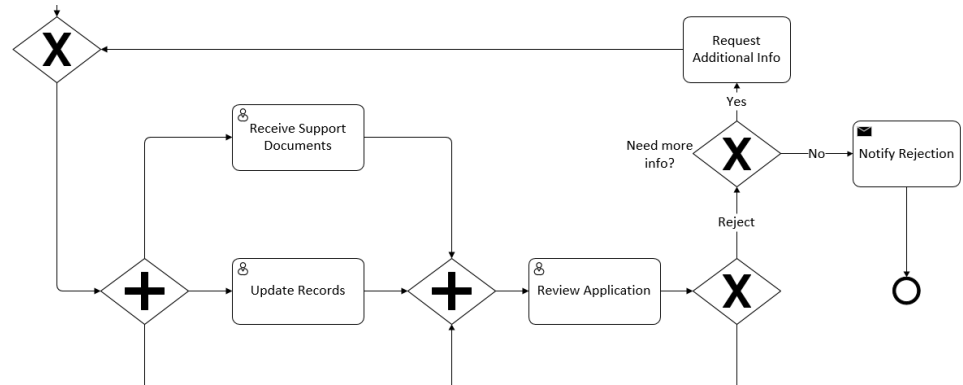
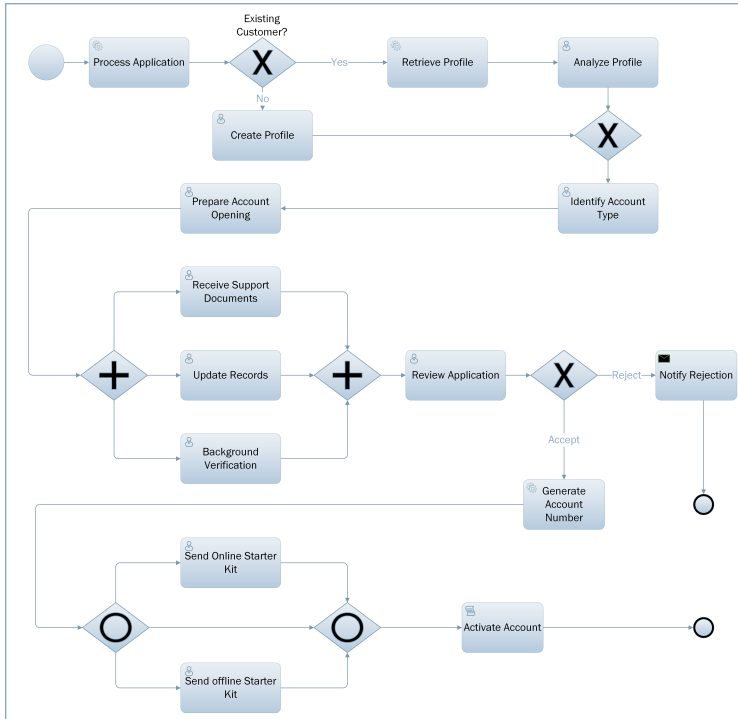
- Up-to-alphabet / up-to-renaming evolution: checking former relations hiding or renaming parts of the alphabet

$$\text{LTS}'(P_1) =_{\text{br}} \text{LTS}'(P_2) \text{ where } \text{LTS}'(P_i) = \text{hide } A \text{ in } \text{LTS}(P_i)$$

- Property preserving evolution: both processes satisfy a same temporal property  $P$

$$\text{LTS}(P_1) \models P \text{ and } \text{LTS}(P_2) \models P$$

# Example (V2)



Conservative evolution ✗

Inclusive evolution ✓

Exclusive evolution ✗

Property preserving evolution « any process execution eventually terminates by a rejection notification or by an account activation » ✗

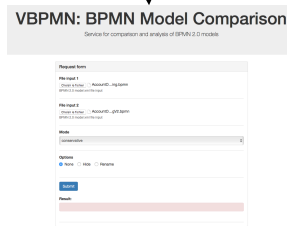


# Outline

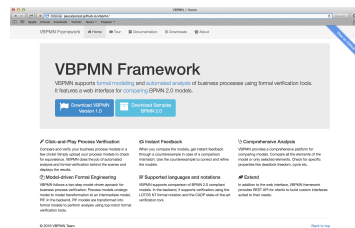
1. BPMN
2. From BPMN to LTS
3. Process Comparison
- 4. Tool Support**
5. Concluding Remarks

# The VBPMN Platform

BPMN 2.0 compliant platforms  
(Activiti, Bonita BPM, ..)



Other workflow-based languages  
(UML activ. diagrams, YAWL, ..)



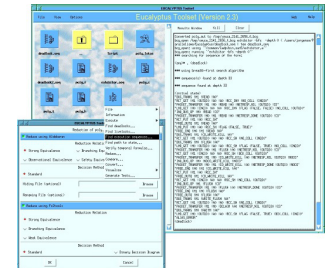
Process  
Intermediate Format

Generated code  
(LNT + SVL)

CADP verification tools

Model-to-model  
Transformation

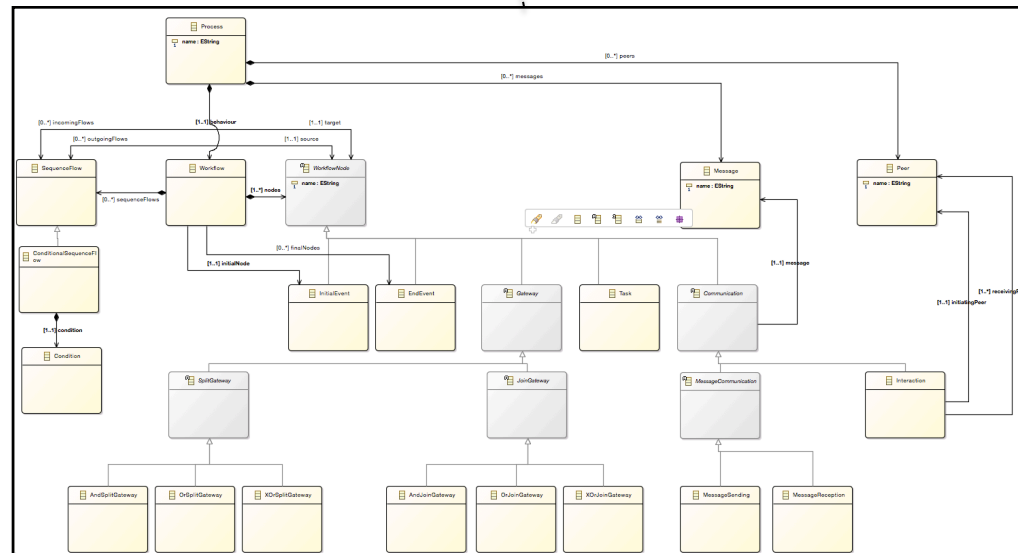
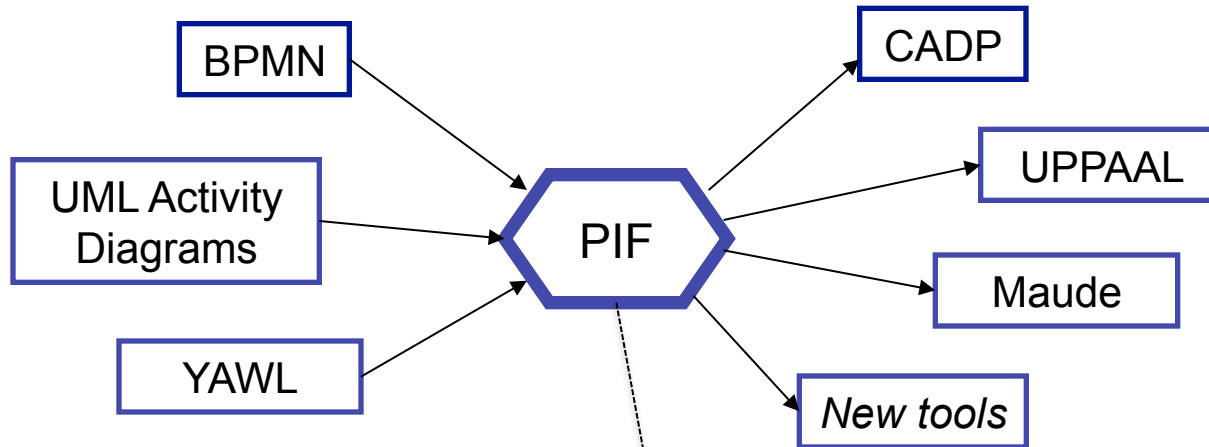
Model-to-text  
Transformation



Other verification techniques  
(formal proof, testing, ..)

Diagnostics

# Process Intermediate Format



# Web Interface

## VBPMN: BPMN Model Comparison

Service for comparison and analysis of BPMN 2.0 models

Request form





















**File input 1**  
 PublishingSystemV3.bpmn  
BPMN 2.0 model xml file input

**File input 2**  
 PublishingSystemV2.bpmn  
BPMN 2.0 model xml file input

**Mode**  
conservative

**Options**  
 None  Hide  Rename

# Experiments

BPMN Proc.	Size			LTS (states/transitions)		Evol. ≡ < >
	Tasks	Flows	Gateways	Raw	Minimized	
1	6	11	2 	29/29	8/9	× √ × 15s
1'	7	15	2  + 2 	78/118	11/14	
2	4	7	1 	70/105	7/9	√ √ √ 15s
2'	8	14	2 	36/38	10/12	
3	7	14	2  + 2 	62/87	10/11	× × × 15s
3'	8	16	4 	1,786/5,346	28/56	
4	15	29	3  + 2  + 2 	469/1,002	24/34	× √ × 15s
4'	16	33	5  + 2  + 2 	479/1,013	26/37	
5	12	24	6 	742,234/3,937,158	148/574	× × √ 31s
5'	12	24	4  + 2 	6,394/21,762	60/152	
6	20	43	6  + 6 	4,488,843/26,533,828	347/1,450	× √ × 9m31s
6'	20	39	8 	4,504,775/26,586,197	348/1,481	

# Outline

1. BPMN
2. From BPMN to LTS
3. Process Comparison
4. Tool Support
5. **Concluding Remarks**

# Concluding Remarks

- We have presented an approach for **automatically checking the evolution of BPMN processes**
- We have defined a **BPMN to LNT translation**, which allows to provide an **LTS semantics for BPMN**
- We have proposed **several notions of evolution** taking inspiration in concurrency theory
- We have implemented our approach in a tool, **VBPMN**, which can be used via a **Web application**
- **Perspectives**: support of unbalanced workflows, extending the BPMN subset considered, quantitative analysis, ...